

WRDC-TR-90-8007
Volume VIII
Part 15

AD-A248 923



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 15 - Forms Editor User's Manual

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209



September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

92-10291

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

92 4 21 131

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

David L. Judson
This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

Bruce A. Rasmussen
BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None			
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UM 620344400		5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VIII, Part 15			
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI			
6c. ADDRESS (City, State, and ZIP Code) 2970 Presidential Drive Fairborn, OH 45324-6209		7b. ADDRESS (City, State, and ZIP Code) WPAFB, OH 45433-6533			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF	8b. OFFICE SYMBOL (if applicable) WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464			
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, Ohio 45433-6533		10. SOURCE OF FUNDING NOS.			
11. TITLE See block 19 Forms E		PROGRAM ELEMENT NO. 78011F	PROJECT NO. 595600	TASK NO. F95600	WORK UNIT NO. 20950607
12. PERSONAL AUTHOR(S) Structural Dynamics Research Corporation: Barker, S. et al.					
13a. TYPE OF REPORT Final Report	13b. TIME COVERED 4 / 1 / 87 - 12 / 31 / 90	14. DATE OF REPORT (Yr., Mo., Day) 1990 September 30	15. PAGE COUNT 253		
16. SUPPLEMENTARY NOTATION WRDC/MTI Project Priority 6203					
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify block no.)			
FIELD 1308	GROUP 0905				
19. ABSTRACT (Continue on reverse if necessary and identify block number) This manual provides application programmers with information to write application programs that use the Form Processor.					
BLOCK 11: INTEGRATED INFORMATION SUPPORT SYSTEM Vol VIII - User Interface Subsystem Part 15 - Forms Editor User's Manual					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT.		21. ABSTRACT SECURITY CLASSIFICATION Unclassified			
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson		22b. TELEPHONE NO. (Include Area Code) (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI		

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

<u>SUBCONTRACTOR</u>	<u>ROLE</u>
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS

	<u>Page</u>	
SECTION 1.0	INTRODUCTION.....	1-1
SECTION 2.0	DOCUMENTS.....	2-1
2.1	Reference Documents	2-1
2.2	Terms and Abbreviations	2-2
SECTION 3.0	ELECTRONIC FORM CHARACTERISTICS.....	3-1
3.1	Fields.....	3-1
3.1.1	Item Fields.....	3-1
3.1.2	Form Fields.....	3-1
3.1.3	Window Fields.....	3-2
3.1.4	Graph Fields.....	3-2
3.1.5	Graphics Fields.....	3-2
3.2	Text.....	3-3
3.3	Interactive.....	3-3
3.4	Attributes.....	3-3
3.4.1	Background.....	3-4
3.4.2	Initial Value.....	3-4
3.4.3	Conversion.....	3-4
3.4.4	Data Type.....	3-4
3.4.5	Entry.....	3-5
3.4.6	Display.....	3-5
3.4.7	Help.....	3-5
3.4.8	Identification.....	3-5
3.4.9	Justification.....	3-5
3.4.10	Location	3-5
3.4.11	Repetition	3-6
3.4.12	Size	3-6
3.4.13	Scroll	3-6
3.4.14	Value Limit	3-6
3.5	Defining Electronic Forms.....	3-6
3.6	Storing Forms in the IISS Environment...	3-6
SECTION 4.0	FORM DEFINITION LANGUAGE.....	4-1
4.1	FDL Format Notation.....	4-1
4.2	Statement Format.....	4-2
4.3	Form Definition Language.....	4-2
4.3.1	Form Definition	4-3
4.3.1.1	CREATE FORM Statement	4-4
4.3.1.1.1	SIZE Clause	4-5
4.3.1.1.2	ATTRIBUTE Clause	4-6
4.3.1.1.3	KEYPAD Clause	4-7
4.3.1.1.4	BACKGROUND Clause	4-8
4.3.1.1.5	PROMPT Clause	4-9

4.3.1.2	Item Field Definition.....	4-10
4.3.1.2.1	ITEM Statement	4-11
4.3.1.2.2	Location Clause	4-12
4.3.1.2.3	SIZE Clause	4-13
4.3.1.2.4	VALUE Clause	4-14
4.3.1.2.5	NODUP Clause	4-15
4.3.1.2.6	DISPLAY AS Clause	4-16
4.3.1.2.7	DOMAIN Clause	4-17
4.3.1.2.8	HELP Clause	4-20
4.3.1.2.9	PROMPT Clause	4-21
4.3.1.2.10	APPEARS IF Clause	4-22
4.3.1.3	Form Field Definition.....	4-23
4.3.1.3.1	FORM Statement	4-24
4.3.1.3.2	Location Clause	4-25
4.3.1.3.3	SIZE Clause	4-26
4.3.1.3.4	PROMPT Clause	4-27
4.3.1.3.5	APPEARS IF Clause	4-28
4.3.1.4	Window Field Definition	4-29
4.3.1.4.1	WINDOW Statement	4-30
4.3.1.4.2	Location Clause	4-31
4.3.1.4.3	SIZE Clause	4-32
4.3.1.4.4	BACKGROUND Clause	4-33
4.3.1.4.5	PROMPT Clause	4-34
4.3.1.4.6	APPEARS IF Clause	4-35
4.3.1.5	Graph Field Definition	4-36
4.3.1.5.1	GRAPH Statement	4-37
4.3.1.5.2	Location Clause	4-38
4.3.1.5.3	SIZE Clause	4-39
4.3.1.5.4	PROMPT Clause	4-40
4.3.1.5.5	APPEARS IF Clause	4-41
4.3.2	Graph Definition	4-42
4.3.2.1	CREATE GRAPH Statement	4-45
4.3.2.2	Location Clause	4-46
4.3.2.3	SIZE Clause	4-47
4.3.2.4	USING Clause	4-48
4.3.2.5	LEGEND Clause	4-49
4.3.2.6	LABEL Clause	4-50
4.3.2.7	ATTRIBUTE Clause	4-52
4.3.2.8	BACKGROUND Clause	4-53
4.3.2.9	AXIS Clause	4-54
4.3.2.9.1	DISPLAY AS Phrase	4-55
4.3.2.9.2	DIRECTION Phrase	4-56
4.3.2.9.3	LABEL Phrase	4-57
4.3.2.9.4	Location Phrase	4-58
4.3.2.9.5	MINIMUM Phrase	4-59
4.3.2.9.6	MAXIMUM Phrase	4-60
4.3.2.9.7	SCALE Phrase	4-61
4.3.2.9.8	SIZE Phrase	4-63
4.3.2.9.9	TICK Phrase	4-64
4.3.2.9.10	GRID Phrase	4-68

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Distr	Avail and/or Special
A-1	

4.3.2.10	CURVE Clause	4-69
4.3.2.10.1	USING Phrase	4-70
4.3.2.10.2	VERSUS Phrase	4-71
4.3.2.10.3	DISPLAY AS Phrase	4-72
4.3.2.10.4	LEGEND Phrase	4-73
4.3.2.10.5	SHADE COLOR Phrase	4-74
4.3.2.10.6	ABSOLUTE Phrase	4-75
4.3.2.10.7	ADDITIVE Phrase	4-76
4.3.2.11	PIE Clause	4-77
4.3.2.11.1	EXPLODE Phrase	4-78
4.3.2.11.2	LABEL Phrase	4-79
4.3.2.11.3	LEGEND Phrase	4-80
4.3.2.11.4	PERCENT Phrase	4-81
4.3.2.11.5	QUANTITY Phrase	4-82
4.3.2.11.6	SHADE COLOR Phrase	4-83
4.3.3	Two Dimensional (2-D) Graphics Field	
	Definitions	4-84
4.3.3.1	Polyline Field Definition	4-85
4.3.3.1.1	POLYLINE Statement	4-86
4.3.3.1.2	STYLE Clause	4-87
4.3.3.1.3	SIZE Clause	4-89
4.3.3.1.4	COLOR Clause	4-90
4.3.3.1.5	POINTS Clause	4-91
4.3.3.1.6	APPEARS IF Clause	4-92
4.3.3.2	Polymarker Field Definition	4-93
4.3.3.2.1	POLYMARKER Statement	4-94
4.3.3.2.2	STYLE Clause	4-95
4.3.3.2.3	SIZE Clause	4-97
4.3.3.2.4	COLOR Clause	4-98
4.3.3.2.5	POINTS Clause	4-99
4.3.3.2.6	APPEARS IF Clause	4-100
4.3.3.3	Graphics Text Field Definition	4-101
4.3.3.3.1	GTEXT Statement	4-102
4.3.3.3.1	EXPANSION Clause	4-103
4.3.3.3.2	SPACING Clause	4-104
4.3.3.3.3	SIZE Clause	4-105
4.3.3.3.4	COLOR Clause	4-106
4.3.3.3.5	PATH Clause	4-107
4.3.3.3.6	G location Clause	4-108
4.3.3.3.7	APPEARS IF Clause	4-109
4.3.3.4	Fillarea Field Definition	4-110
4.3.3.4.1	FILLAREA Statement	4-111
4.3.3.4.2	COLOR Clause	4-112
4.3.3.4.3	POINTS Clause	4-113
4.3.3.4.4	APPEARS IF Clause	4-114
4.3.4	Location Clause/Parameter	4-115
4.3.4.1	Absolute Location	4-117
4.3.4.2	Relative Location	4-118
4.3.4.2.1	Relative Location (Above/Below)	4-119
4.3.4.2.2	Relative Location (Right/Left)	4-120
4.3.4.2.3	Location Relative to Two Fields	4-121
4.3.4.3	Combination Location	4-123

4.3.5	G_location Clause/Parameter	4-126
4.3.6	Repeat Spec Parameter	4-128
4.3.6.1	One Dimensional Array	4-129
4.3.6.2	Two Dimensional Array	4-129
4.3.6.3	Three Dimensional Array	4-130
4.3.6.4	One Dimensional Scrolled Array	4-130
4.3.6.5	Two Dimensional Scrolled Array	4-131
4.3.7	Expression Parameter	4-132
4.3.8	Non-Graphics Primitives	4-142
4.3.9	Graphics Primitives	4-144
4.3.10	Pre-defined Attributes	4-147
4.4	Qualified Names	4-147
4.5	Restrictions	4-148
4.6	Abbreviations	4-148
4.7	Including Comments	4-148
4.8	Reserved Words	4-148
SECTION	5.0 FORM DRIVEN FORM EDITOR	5-1
5.1	FDFE Functional Organization	5-1
5.2	FDFE Editing Features	5-2
5.3	System Operation	5-3
5.3.1	Accessing the FDFE	5-3
5.4	Work Task Screen	5-4
5.4.1	Choosing a Work Task	5-4
5.4.2	LS (List FDL Sources)	5-6
5.4.2.1	List FDL Source Files Screen	5-8
5.4.3	IS (Insert FDL Source)	5-8
5.4.4	MS (Modify FDL Source)	5-9
5.4.5	SS (Select FDL Source)	5-11
5.4.6	CS (Copy FDL Source)	5-12
5.4.7	RS (Rename FDL Source)	5-13
5.4.8	DS (Drop FDL Source)	5-14
5.4.9	LC (List Compiled form definitions)	5-15
5.4.9.1	List of Compiled Form Definitions Screen	5-16
5.4.10	VC (View Compiled form definition)	5-16
5.4.11	DC (Drop Compiled form definition)	5-18
5.4.12	EX (EXit form driven form editor)	5-19
5.5	Edit Task Screen	5-19
5.5.1	Choosing an Edit Task	5-20
5.5.2	LF (List Forms)	5-20
5.5.2.1	List of Forms Screen	5-22
5.5.3	WF (Write Form)	5-22
5.5.4	CF (Compile a Form)	5-24
5.5.5	SF (Select a Form)	5-25
5.5.5.1	Using Single Field Mode to Review a Form	5-26
5.5.5.2	Using Form Mode to Review a Form	5-29
5.5.5.3	Using Layout Mode to Review a Form	5-34
5.5.6	IF (Insert a Form)	5-34
5.5.6.1	Using Single Field Edit Mode to Define a Form	5-35
5.5.6.1.1	Form Area Fields	5-36
5.5.6.1.2	Required Field Information	5-37
5.5.6.1.3	Optional Field Information	5-39
5.5.6.1.4	Item Only Information	5-40

5.5.6.2	Using Form Mode to Define a Form	5-42
5.5.6.3	Using Layout Mode to Define a Form	5-46
5.5.6.3.1	Layout Mode Symbols	5-47
5.5.6.4	Using Icon Mode to Define an Icon	5-48
5.5.6.4.1	Creation of an Object	5-49
5.5.6.4.2	Selection of an Object	5-51
5.5.6.4.3	Overlapping Objects	5-53
5.5.6.4.4	Moving and Scaling of an Object	5-54
5.5.6.4.5	Moving and Scaling of an Icon	5-56
5.5.6.4.6	Copy Selected Object	5-58
5.5.6.4.7	Rotating	5-58
5.5.6.4.8	Changing Color	5-58
5.5.6.4.9	Deleting an Object	5-58
5.5.6.4.10	Deleting an a Polyline Coordinate	5-58
5.5.6.4.11	Inserting an a Polyline Coordinate	5-59
5.5.6.4.12	Changing Line Style	5-59
5.5.6.4.13	Keypad Help	5-59
5.5.7	MF (Modify a Form)	5-59
5.5.7.1	Using Single Field Mode to Modify a Form	5-60
5.5.7.2	Using Form Mode to Modify a Form	5-63
5.5.7.3	Using Layout Mode to Modify a Form	5-65
5.5.7.4	Using Icon Mode to Modify an Icon	5-66
5.5.8	DF (Drop a Form)	5-67
5.5.9	EW (Exit Write)	5-68
5.5.1	EC (Exit Compile)	5-68
5.5.1	EN (EXIT NO SAVE)	5-69
5.6	Limited Edit Task Screen	5-70
SECTION 6.0	FDL COMPILER - FLAN	6-1
6.1	Executing Flan	6-1
6.1.1	FLAN in the IISS Environment	6-1
6.1.2	FLAN as a Batch Program	6-2
6.2	FLAN Error Messages	6-2
6.2.1	Warning Message	6-2
6.2.2	Error Messages	6-2
6.2.3	Fatal Messages	6-4
SECTION 7.0	FORM TOOLS	7-1
7.1	Generating Include Members	7-1
SECTION 8.0	SAMPLE FORM DEFINITION	8-1

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
		3-1
4-1	IISS Form Storage Hierarchy	3-7
4-2	Field Reference Points	4-116
4-3	Absolute Location	4-117
4-4	Relative Location	4-119
4-5	Relative Location (Above/Below)	4-120
4-6	Relative Location (Right/Left)	4-121
4-7	Location Relative to Two Fields	4-122
4-8	Absolute Row/Relative Column	4-123
4-9	Absolute Column/Relative Row	4-124
4-10	One Dimensional Array	4-129
4-11	Two Dimensional Array	4-129
4-12	Three Dimensional Array	4-130
4-13	One Dimensional Scrolled Array	4-131
5-1	Two Dimensional Scrolled Array	4-131
5-1	FDFE Functions Mapped to the IISS Form Storage Hierarchy	5-2
5-2	VT100 Keypad Layout for the FDFE Application	5-3
5-3	Work Task Screen	5-4
5-4	Choosing a Work Task - Menu Selection	5-5
5-5	Choosing a Work Task - Command Entry	5-6
5-6	Choosing LS Using Menu Selection	5-7
5-7	List FDL Source Files Screen	5-8
5-8	Choosing IS Using Manual Selection	5-9
5-9	Choosing MS Using Menu Selection	5-10
5-10	Choosing SS Using Menu Selection	5-11
5-11	Choosing CS Using Menu Selection	5-12
5-12	Choosing RS Using Menu Selection	5-13
5-13	Choosing DS Using Menu Selection	5-14
5-14	Choosing LC Using Menu Selection	5-15
5-15	List Compiled Form Definitions Screen	5-16
5-16	Choosing VC Using Menu Selection	5-17
5-17	Choosing DC Using Menu Selection	5-18
5-18	Edit Task Screen	5-19
5-19	Choosing LF Using Menu Selection	5-21
5-20	List of Forms Screen	5-22
5-21	Choosing WF Using Menu Selection	5-23
5-22	Choosing CF Using Menu Selection	5-24
5-23	Choosing SF Using Menu Selection	5-26
5-24	Reviewing a Form in Single Field Mode	5-27
5-25	Reviewing a Form in Form Mode	5-29
5-26	FIELD CHARACTERISTICS TABLE - Part One	5-30
5-27	FIELD CHARACTERISTICS TABLE - Part Two	5-31
5-28	FIELD CHARACTERISTICS TABLE - Part Three	5-32
5-29	FIELD CHARACTERISTICS TABLE - Part Four	5-33

5-30	Reviewing a Form in Layout Mode	5-34
5-31	Choosing IF Using Menu Selection	5-35
5-32	Defining a Form in Single Field Mode	5-36
5-33	Form Mode Screen	5-43
5-34	Defining a Form in Form Mode	5-45
5-35	Layout Mode Screen	5-46
5-36	Defining a Form in Layout Mode	5-48
5-37	Icon Mode Screen	5-49
5-38	Creation of Circle and Arch Primitives	5-50
5-39	Creation of Box and Polyline Primitives	5-50
5-40	Polyline Primitive	5-51
5-41	Box Primitive	5-52
5-42	Selected Circle Box and Polyline	5-52
5-43	Overlapping Objects	5-54
5-44	Moving and Scaling an Object	5-55
5-45	Moving and Scaling an Icon	5-57
5-46	Choosing MF Using Menu Selection	5-60
5-47	Single Field Mode Screen	5-61
5-48	Using Single Field Mode to Delete a Field	5-62
5-49	Form Mode Screen	5-64
5-50	Using Form Mode to Delete a Field	5-65
5-51	Modifying a Form in Layout Mode	5-66
5-52	Modifying an Icon in Icon Mode	5-67
5-53	Choosing DF Using Menu Selection	5-68
5-54	Limited Edit Task Screen	5-70
8-1	Sample Form	8-1

SECTION 1

INTRODUCTION

The Form Editor is a high level utility for defining and maintaining electronic forms. In the User Interface environment, these forms are used to communicate between an application program and the user through calls to the Form Processor. The Form Processor is the IISS run-time package of routines that an application program uses to manipulate and display forms.

This manual is intended for application programmers who write application programs that use the Form Processor and for anyone who creates electronic forms with the Form Definition Language or the Form Driven Form Editor. Knowledge of the Integrated Information Support System (IISS) environment is assumed.

In this manual the term "user" refers to the person running an application program with an electronic forms interface.

SECTION 2

DOCUMENTS

2.1 Reference Documents

- [1] Structural Dynamics Research Corporation, IISS Forms Language Compiler Development Specification, DS 570144401A, 31 May 1988.
- [2] Structural Dynamics Research Corporation, IISS Form Driven Form Editor Development Specification, DS 570144402A, 31 May 1988.
- [3] ICAM Documentation Standards, 15 September 1983, IDS150120000C.

This manual is one of a set of user manuals that together describe how to operate in the IISS environment. The complete set consists of the following manuals listed here for reference:

- [1] Structural Dynamics Research Corporation, IISS Form Editor User's Manual, UM 570144400A, 31 May 1988.
Explains how to define and maintain electronic forms. It is intended to be used by programmers writing application programs that use the Form Processor.
- [2] Structural Dynamics Research Corporation, IISS Form Processor User's Manual, UM 570144200A, 31 May 1988.
Describes the set of callable, execution-time routines available to an application program to process electronic forms. It is intended for use by programmers writing application programs for the IISS environment.
- [3] Structural Dynamics Research Corporation, IISS Terminal Operator Guide, OM 570144000A, 31 May 1988.
Explains how to operate the generic IISS terminal when running an IISS application program. The IISS end user environment function selection and some predefined applications are also described.
- [4] Structural Dynamics Research Corporation, IISS Text Editor User's Manual, UM 570144600A, 31 May 1988.
Explains how to use the file editing functions including: inserting, deleting, moving and replacing text.
- [5] Structural Dynamics Research Corporation, IISS Application Generator User's Manual, UM 570144502A, 31 May 1988.

Describes the Application Definition Language and the process used for translating textual definitions of interactive database applications into programs that access selected data base information resident in the Common Data Model. This information is accessible through the IISS Neutral Data Manipulation Language.

- [6] Structural Dynamics Research Corporation, IISS Virtual Terminal User's Manual, UM 570144300A, 31 May 1988.
- Explains the program callable interface to the IISS Virtual Terminal. The callable routines, Virtual Terminal commands and the implementation of additional terminals are described. It is intended for application and system programmers working in the IISS environment.
- [7] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 570144200, 31 May 1988.
- [9] Structural Dynamics Research Corporation, C Coding Guidelines,
- [10] American National Standards Institute, Computer Graphics - Graphical Kernel System (GKS) Functional Description, ANSI X3.124-1985, 24 June 1985.
- [11] American National Standards Institute, Computer Graphics - Metafile for the Storage and Transfer of Picture Description Information, ANSI X3.122-1986, 27 August 1986.
- [12] American National Standards Institute, Additional Controls for use with American National Standard Code for Information Interchange, ANSI X3.64-1975.

2.2 Terms and Abbreviations

Abbreviation id: A user-defined abbreviation of a database table that may be used as a qualifier for a data item instead of the table name itself.

American Standard Code for Information Interchange (ASCII): The character set defined by ANSI X3.4 and used by most computer vendors.

Application Interface (AI): A subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Definition Language (ADL): A subset of the Form Definition Language (FDL) which provides for the definition

of interactive, form-based applications. ADL is part of FDL. The statements are included in the same source file as the form definitions.

Application Generator (AP): A translator which accepts ADL as input and generates an interactive application as output.

Application id: A user-defined name which is used as the root of the file name of the generated application.

Application Process (AP): A cohesive unit of software that can be initiated as a unit to perform some function or functions.

Attribute: A field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms and window fields only. Foreground attributes are defined for item fields. Attributes may be permanent (they remain the same unless changed by the application program), or they may be temporary (they remain in effect from the application's first call to OISCR after a call to PUTATT or PUTBAK to the next call to OISCR).

Background Attribute: A color which is applied to the background of forms, graphs, window fields, and item fields. Specifying a background attribute is analogous to specifying what color of paper to print a form, etc. on.

Bar Graph: A graph which represents dependent data values correlated to an independent variable by horizontal or vertical rectangles which are proportional to the dependent data values.

C Code: A user-defined sequence of C language statements which are to be included in the generated application.

Closed Figure: A figure is closed if the path traced by a moving point returns to its starting position. The starting position may be arbitrarily assigned. "Fillarea" is synonymous with "closed figure".

Complex Figure: A figure is complex if the path traced by a moving point crosses itself. An arbitrary point may be determined to be contained within the traced boundary if a line drawn to infinity crosses the boundary an odd number of times. If the number of crossings is zero or even, the point is outside of the traced boundary.

Condition Action: An action to be taken when a specified condition is true.

Condition Definition: Specifies pre-defined actions that will occur as the result of user interaction with the software via the terminal.

Dependent Data: Data correlated to a dependent variable.

Dependent Variable: A mathematical variable whose value is determined by that of one or more other variables in a function.

Device Drivers (DD): Software modules written to handle I/O for a specific kind of terminal. The modules map terminal specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: A list of all the open forms that are currently being processed by the Form Processor or the user.

Element: A graphics line or other primitive composed of graphics lines, such as an arc.

Extended Binary Coded Decimal Interchange Code (EBCDIC): The character set used by a few computer vendors (notably IBM) instead of ASCII.

Field: An area on a form which is a place-holder for data. A field may be an item, a form, a window, or a graph.

Field Name: A qualified name enclosed in single quotes which denotes a field in the Form Processor display list.

Figure: A collection of elements. A figure may be closed or open.

Fillarea: A geometric primitive consisting of a planar area which is to be filled in with a particular color or pattern. A fillarea must be closed.

Flag id: A user defined name which contains a boolean valued state indicator which may be used in condition and set expressions.

Foreground Attribute: A characteristic which is applied to the text of a form, graph, or item. Specifying a background attribute is similar to specifying what color of ink to use to print a form on paper. Foreground attributes may also include such characteristics as blinking, highlighted, etc.

Form: A structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, windows, and graphs.

Form Definition (FD): The object file produced by the compilation of a Form Definition Language source file. It is read at run-time by the Form Processor.

Form id: A name which denotes a form specified in an FDL file.

Form Definition Language (FDL): The language in which electronic forms and the reports and applications which interact with them are defined.

Form Driven Form Editor (FDFE): A subset of the FE which consists of a form driven application used to create Form Definition files interactively.

Form Editor (FE): A subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Form Driven Form Editor and the Forms Language Compiler.

Form Field: A place-holder on a form for a subform which is to be supplied in the form definition. A subform should contain a group of logically related data. A form field is static, that is, each time the form which contains the form field appears, the same subform will be displayed in the form field.

Form Hierarchy: A graphic representation of the way in which forms, items and windows are related to their parent form.

Form Processor (FP): A subset of the IISS User Interface that consists of a set of callable execution-time routines available to an application program for form processing.

Form Language Compiler (FLAN): A translator that accepts a series of Form Definition Language statements and produces form definition files as output.

Form Processor Text Editor (FPTE): A subset of the Form Processor that consists of software modules that provide text editing capabilities to all users of applications that use the Form Processor.

Graph: A picture correlated with data that alters as the data changes; by necessity, this is a dynamic (not pre-defined) picture. A graph may be imposed on windows or forms.

Graph Field: An area on a form into which a pre-defined graph will be placed at run-time.

Graph Figure: A collection of graphics primitives. The primitives can be circles, lines, arcs, etc.

Graphics Kernel System (GKS): A two-dimensional graphics standard which is defined independently of any programming language.

Icon: A collection of figures and points that is pre-defined. An icon may be imposed on windows and forms. "Icon" is synonymous with "picture".

IISS Function Screen: the first screen that is displayed after logon. It allows the user to specify the function he wants to access and the device type and device name on which he is working.

Independent Data: Data that is correlated to an independent variable.

Independent Variable: A mathematical variable whose value is specified first and determines the value of one or more other values in an expression or function. For example, In a business graph of sales versus month, month is the independent variable and sales is the dependent variable, because sales varies by month.

Integrated Information Support System (IISS): A test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item Field: A non-decomposable area of a form in which text may be placed and the only defined areas where user data may be input/output.

Item Name: A qualified name which denotes an item field.

Key id: A user-defined name for a function key which is how the key is referenced within an ADL program.

Line Graph: A graph which represents dependent data values correlated to an independent variable by points connected with a broken line.

Message: Descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: A line on the terminal screen that is used to display messages.

Network Transaction Manager (NTM): The IISS subsystem which performs the coordination, communication and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Open Figure: A figure is open if the path traced by a moving point does not return to its starting position. The starting position may be arbitrarily assigned. "Polyline" is synonymous with "open figure".

Open List: A list of all the forms that have been and are currently open for an application process.

Operand: A value which is operated on by some function.

Operating System (OS): Software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Operator: A mathematical or logical symbol denoting an operation to be performed (e.g., +, -, /, *, etc.).

Page: An instance of a form in a window that is created whenever a form is added to a window.

Paging and Scrolling: A method which allows a form to contain more data than can be displayed at one time with provisions for viewing any portion of the data buffer.

Parameter form id: The name of a form which is to be displayed when the application is started from the IISS function screen and which allows the user to specify run-time values for the application.

Physical Device: A hardware terminal.

Picture: A collection of figures and points that is pre-defined. A picture may be imposed on a window or a form. "Picture" is synonymous with "icon".

Pie Graph: A graph which represents data values on a circle cut into sections by radii. Each section of the circle represents a data value as a percentage of the total of all the values represented by the graph.

Point: A marker or a symbol.

Polyline: A geometric primitive consisting of one or more connected line segments.

Polymarker: A geometric primitive consisting of one or more marker symbols (such as a cross or a dot).

Procedure id: The name of a user-supplied or system service procedure which is to be called by the generated application.

Primitive: The lowest level of definition for an attribute. Primitives define the characteristics of an attribute.

Qualified Name: The name of a form, item, window, or graph preceded by the hierarchy path so that it is uniquely identified.

Subform: A form that is used within another form.

Text: A geometric primitive consisting of a number of characters with a particular orientation arranged along a particular path and aligned in a particular manner with some point.

User Data: Data which is either input by the user or output by the application programs to items.

User Interface (UI): The IISS subsystem which controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System (UIDS): The collection of IISS User Interface subsystems that is used by application programmers as they develop IISS applications. The UIDS includes the Form Editor and the Application Generator.

User Interface Management System (UIMS): The run-time UI. It consists of the Form Processor, Virtual Terminal, Application Interface, the User Interface Services and the Text Editor.

User Interface Monitor (UIM): The part of the Form Processor which handles messaging between the NTM and the UI. It also provides authorization checks and initiates applications.

User Interface Services (UIS): The subset of the IISS User Interface that consists of a package of routines that aid users in controlling their environment. It includes message management, change password, and application definition services.

User Interface/Virtual Terminal Interface (UI/VTI): Another name for the User Interface.

Virtual Terminal (VT): The subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.

Window: A dynamic area of a terminal screen on which pre-defined forms may be placed at run-time.

Window Field: A place-holder on a form for subforms which are to be supplied at run-time. A window field is dynamic, that is, based upon data which may change, a different subform may be placed in the window each time the containing form appears.

Window Manager: A facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor.

Window Name: A qualified name which denotes a window.

SECTION 3

ELECTRONIC FORM CHARACTERISTICS

Electronic forms are very similar to the paper forms we encounter in our everyday lives. They each contain two types of information. First, the forms have information already printed on them - titles, headings, descriptions of items to be filled in, and instructions. Second, the forms have blank spaces for information, such as name, date, and so on, to be filled in by the users.

This User's Manual is concerned with the design and definition of electronic forms. The first step is to examine the individual components of a form.

3.1 Fields

Electronic forms consist of areas called fields which allow the user to enter and view variable data. A field on an electronic form must be one of the following:

- o An item
- o A form
- o A window
- o A graph
- o Or one of the following four types of graphics fields:
 - Polyline
 - Polymarker
 - Graphics Text
 - Fillareas

3.1.1 Item Fields

The most common type of field is an item field. An item field is an area on a form reserved for a single piece of data. For example, the area on a form into which you might be asked to enter the current date will be defined as an item field.

3.1.2 Form Fields

In many applications, there is a large volume of information to be communicated. This can result in many forms and also forms with many items. As the number of forms and item fields per form increases, there may be groups of items that are shared by more than one form or that make sense by themselves. For example, all IRS forms require identification information such as the name, address, and social security number of the taxpayer. These logical groups of items can be made into separate forms and incorporated into other forms as subforms. To incorporate a subform within a form, the area on the host form where the group of items should be located is defined as a form field. A subform within a form is similar to using macros or procedures in a programming language. Forms can be nested to any level that makes sense to keep the form definition for an application as simple and straightforward as possible.

3.1.3 Window Fields

Although the form within a form concept provides a great deal of flexibility in defining forms, these form arrangements are static and cannot be changed. Often the information a user needs to enter for an application is dependent on information that was previously entered. For example, in a part inventory application, you may want to continue displaying the same part identification information while requesting other types of information. Defining a field on a form as a window provides this capability. A window is used as a place holder on the form. At run-time, the application then determines what forms or graphs the window will contain. A window field can contain more than one form at a time although, in general, only one form will be visible. Each form or graph added to a window creates a page and just as with a book, unless the page is a transparent overlay, following pages cannot be seen.

3.1.4 Graph Fields

A "graph field" must first be distinguished from a "graph". A "graph field" is an area on a form which is reserved for displaying a graph, just like a form field reserves an area on a form for displaying another form. A graph is the actual graph being displayed. A graph may either be drawn into a form by use of the graph field, much like a form field draws in a subform, or it may be presented directly into a window, just like a form can be presented in a window. Viewed in this context, a graph is a specialized form. A graph field contains only one graph. The graph must be defined separately. Like an item field, a graph also occupies the lowest level of the hierarchy. That means that a graph cannot contain items or other graphs. The user must be careful to distinguish between a "graph" and a "graph field".

3.1.5 Graphics Fields

A graphics field is an area on a form reserved for a graphics primitive. Each type graphics field can hold only one type of primitive. When the graphics field is defined, you must define the type of primitive the field will contain.

The primitives provided are polylines, polymarkers, fillareas, and graphics text. These primitives may be combined to create a picture on the form.

Graphics characters differ from ordinary characters in that they can be manipulated as to size, orientation, and color.

Graphics fields differ from graph fields in that a graph field contains the entire picture (the graph), whereas the graphics field contains only one primitive and must be combined with other graphics fields to create a picture. A graph is dynamic, changing as the data which is represented changes. Graphics fields are static, that is, they do not change their

size and shape at run-time. They may or may not appear on the form if the Appears If capability is used, but the size and shape will not change.

In summary, windows can contain forms or graphs, and forms can contain items, forms, windows, graphs, and graphics primitives. Graphs cannot contain items, forms, or other graphs. The Form Processor User's Manual explains how to create and use window pages for an application program.

3.2 Text

The information which is displayed on an electronic form is called text. Text must be distinguished from data which is contained in a field on a form. Text is displayed directly on the form. Data is displayed in a field on a form.

There are two types of text on any form. They are prompt text and background text. Prompt text is text which is associated with a field and is positioned relative to it. It may be used to tell the user what to enter in a particular field, or to describe the information which the application may place in the field. In the following example, "DATE:" is prompt text:

DATE: _____

Background text is not related to any given field and is used primarily for titles, headers, and formatting. For example, the words "Please print" at the top of a form would be background text. A line of asterisks used to separate a form into sections or to draw a box around a set of fields is also be background text.

3.3 Interactive

The ability to interact with a machine in a user-friendly environment is a very important factor to consider when developing a high quality user interface. This user-friendly environment can be provided in part by giving the user a comfortable, non-intimidating means of interaction. Electronic forms have many advantages over paper forms, one of which is the ability to interact with the user in a helpful manner. This interaction can provide the user with help in filling out forms by prompting for information, providing feedback about the "correctness" of that information once it has been entered, or providing detailed explanations at the user's request.

3.4 Attributes

To a user running an IISS application, a form is a collection of information displayed on the terminal screen. To the User Interface, a form is a data structure that specifies how information is to be displayed on a terminal and how the user can interact with that display. Attributes are the

characteristics of electronic forms that specify this information. The complete list of electronic form attributes is described next.

3.4.1 Background

The background attribute applies to forms, windows, and graphs. It is analogous to printing paper forms on different colors of paper. In addition to displaying white characters on a black background or black characters on a white background (reverse video), you have the option of specifying a variety of background colors or defining a specific background attribute for your application.

3.4.2 Initial Value

The initial value attribute applies to item fields. Items can be initialized so that they contain a value when the containing form is first displayed, instead of appearing blank. This feature is particularly useful for specifying default values or for suggesting an appropriate entry to a user. If no initial value is given explicitly, the default initial value is a string of blanks equal in length to the size of the item.

Initial values are valid for all items regardless of their display attributes. If the user does not replace an initial value in an input item, the initial value is used as the user's input just as if the value has been keyed in. The initial value for an item can be:

- o An integer
- o A string constant
- o The result of evaluating an expression
- o The value of another item field
- o The first displayed element of a specified array
- o A built-in function such as "date" or "time"

3.4.3 Conversion

The conversion attribute causes case conversion of a data value that has been entered in an item field. The value can be converted to all lower-case or upper-case alphabetic characters. Specifying this attribute is optional. If this attribute is not specified, the data value remains as it was entered by the user.

3.4.4 Data Type

The data type attribute is used to determine the type of data that can be entered in an item field. If the numeric option is specified, the value entered for the item must contain all numbers. If numeric is not specified, the default data type is alphanumeric. If a value limit attribute is specified, numeric is automatically assumed.

3.4.5 Entry

The entry attribute specifies entry requirements for item fields. You can specify that a value must be entered for the item before the form can be processed by the application program. You can also specify that the value must fill every position of the item.

3.4.6 Display

The display attribute controls access to an item field and how it appears to the user. If an item is input, the user may enter a value for the item. You can specify whether the user's input will or will not be echoed on the screen. An output item means that only the application program may enter a value for it. You can also specify that an output item can be tabbed to or that only the Form Processor can enter a value for it. One of these options must be specified but it can be changed or temporarily overridden using the FP routine PUTATT as explained in the Form Processor User's Manual.

3.4.7 Help

The help attribute provides user assistance for entering data values for item fields. Help can be either a string or a form that is displayed when the cursor is in the item field and the <HELP> key is pressed or you can specify that the application program decides what to do when the <HELP> key is pressed. How to use the <HELP> function key is explained in the IISS Terminal Operator Guide.

3.4.8 Identification

The identification attribute is a unique name that identifies a form or field. These names are used by the application program to access form data as explained in the Form Processor User's Manual.

3.4.9 Justification

The justification attribute positions an entered data value in an item field. This attribute affects the internal representation of the value as well as how it appears to the user when displayed. A value can begin in the leftmost position of the field or end in the rightmost position of the field. Specifying this attribute is optional. If this attribute is not specified, the position of the data value remains as it was entered by the user.

3.4.10 Location

The location attribute specifies where text and fields will be positioned on the form. Location may be absolute with respect to the origin of the form or relative to other fields on the form.

3.4.11 Repetition

The repetition attribute specifies that the field appears on the form more than once to create arrays of fields.

3.4.12 Size

The size attribute determines the area a field occupies on the form or the area a subform occupies in a form or window field.

3.4.13 Scroll

The scroll attribute allows a form to contain more data than can be displayed at one time. The user can go into the scroll/page mode of the keyboard as explained in the IISS Terminal Operator Guide and press function keys to view different portions of the data in the display area. Scrolling is especially useful for those applications which must display a list of objects from a database or file and the list is larger than the area available for display.

3.4.14 Value Limit

The value limit attribute sets limits on the values that can be entered in a numeric item field. A maximum or minimum value can be set as well as a range of allowable values. Specifying this attribute is optional. If this attribute is specified, the data type is assumed to be numeric.

3.5 Defining Electronic Forms

The Form Editor provides two methods for defining electronic forms. They are the Form Driven Form Editor (FDFE) and the Form Definition Language (FDL).

THE FDFE is an interactive utility that allows the user to define a form by graphically positioning fields and prompts on the terminal screen, by entering values in response to prompts for the necessary information, or by retrieving and copying the necessary information from existing form definitions.

FDL is a high-level programming language which allows the user to key in the form definition through any standard text editor.

3.6 Storing Forms in the IISS Environment

All form definitions are stored in FDL source files. An FDL source file can contain more than one form definition. The FDL compiler (FLAN) must then be used to convert the form definitions into a format understood by the Form Processor. FLAN reads an FDL source file and creates a separate compiled form for each form definition contained in the source file. A compiled form is referred to as an FD file. A hierarchy is implied by this as shown in Figure 3-1.

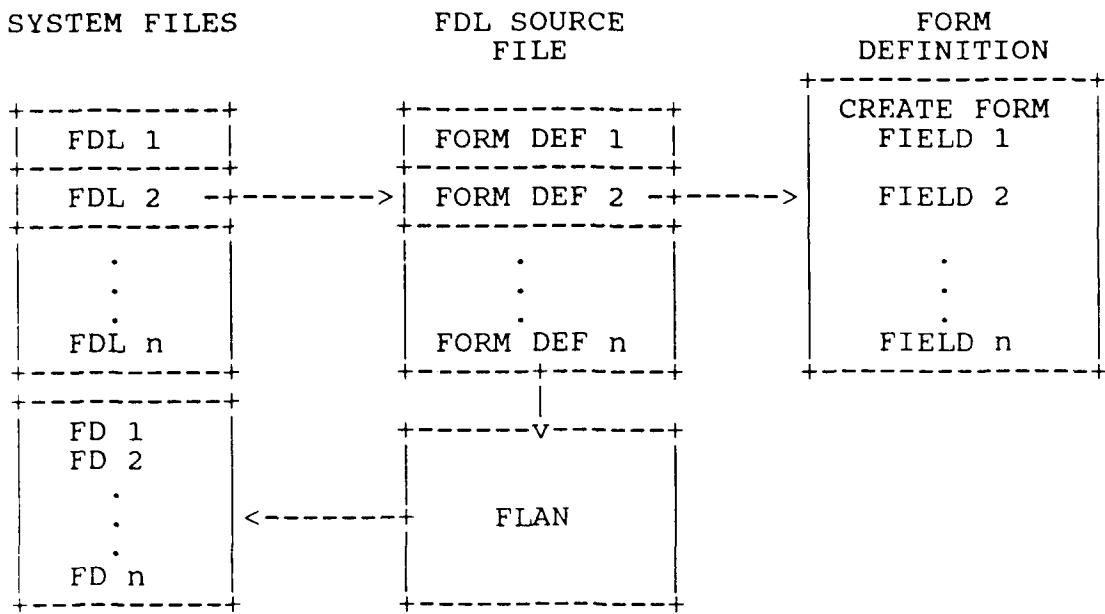


Figure 3-1 IISS Form Storage Hierarchy

Note that the FDL source files which result from forms defined using the FDFE can be edited using any text editor and syntactically correct form definitions created by writing FDL entries directly into an FDL source file can be operated on by the FDFE, as long as the form definitions do not contain graph for graphics fields. The FDFE does not handle business graphs or graphic fields.

SECTION 4

FORM DEFINITION LANGUAGE

The Form Definition Language (FDL) provides a very precise and flexible method for defining electronic forms. Form definitions are created by writing FDL entries directly to an FDL source file with any text editor you might use to prepare a program source file. More than one form may be defined in the same FDL source file.

4.1 FDL Format Notation

This manual uses the following notation to describe the syntax of the FDL language:

UPPER-CASE	identifies reserved words that have specific meanings in the FDL. These words are generally required unless the portion of the statement containing them is itself optional.
lower-case	identifies names, numbers, or character strings that the user must supply.
Initial upper-case	identifies a statement or clause that is defined later on.
_ Underscores	identify reserved words or portions of reserved words that are optional.
{ } Braces	enclosing vertically stacked options indicate that one of the enclosed options is required.
[] Brackets	indicate that the enclosed clause or option is optional. When two or more options are vertically stacked within the brackets, one or none of them may be specified.
... Ellipsis	indicates that the preceding statement or clause may be repeated any number of times.
' ' Single Quotes	indicates that the literal representation of the element's characters is to be used.
_id	a suffix which specifies an identifier.
_name	a suffix which specifies a qualified name.
string	specifies a character string.
int	specifies an integer.
real	specifies a real number.

All other punctuation is to be considered part of the language.

4.2 Statement Format

FDL statements can be entered in free format. Free format means that keywords and numbers can be separated by any number of spaces. The compiler treats tabs, comments, and carriage returns as spaces.

Form and field definition clauses may be in any order.

The DOMAIN clause options may be entered in any order.

Form field names used in the application definition syntax are pseudo qualified names in the form hierarchy. They are denoted by strings of the following type: 'form.item'. The single quotes are required. (Refer to section 4.4 for more information on qualified names).

4.3 Form Definition Language

The Form Definition Language (FDL) provides a very precise and flexible method for defining electronic forms. Form definitions are created by writing FDL entries directly to an FDL source file with any text editor you might use to prepare a program source file. More than one form may be defined in the same FDL source file.

Form Definition

4.3.1 Form Definition

The collection of FDL statements that define a form is a form definition. A form definition is written to an FDL source file with any text editor you might use to prepare a program source file. An FDL source file may contain more than one form definition. The syntax for a form definition is:

```
CREATE FORM form_id  
      [ SIZE int-1 [ BY int-2 ] ]  
      [ ATTRIBUTE attribute_id ( Primitive [ ,... ] ) ] ...  
      [ KEYPAD ( { function_id = int } ... ) ]  
      [ BACKGROUND attribute_id ]  
      [ PROMPT Location prompt_string ] ...  
      [ Field_Definition ] ...
```

CREATE FORM Statement

4.3.1.1 CREATE FORM Statement

The first statement of every form definition must be the CREATE FORM statement. This statement tells the compiler that what follows is a form definition. It also signals the end of the previous form definition if this is not the first one in the source file. The syntax for the CREATE FORM statement is:

CREATE FORM form_id

CREATE FORM is the statement keyword.

form_id is a unique name of up to 10 letters and/or numbers associated with each form. It is used by the application program to identify the form. A form_id is required and cannot begin with a number. The length of this name may be further restricted by the system you are running on. You should check with your system manager for any further restrictions.

CREATE FORM Statement, SIZE Clause

4.3.1.1.1 SIZE Clause

This clause determines the number of columns and rows the form will occupy when it is displayed. This is specified by the width and height. This clause is optional. The syntax for the SIZE clause is:

SIZE int-1 [BY int-2]

SIZE is the clause keyword.

int-1 is the width of the form. It is expressed as the number of columns it will occupy when displayed.

BY is a reserved word that must be included when int-2 is specified. There must be a space before and after this word when used.

int-2 is the height of the field. It is expressed as the number of rows it will occupy when displayed. This parameter is optional and defaults to one if not entered.

The width and/or height values may be specified as "*" to indicate that the size may change at run-time to accommodate changes in the size and location of fields on the form. If you do not specify the size of the form, it defaults to the size of the form or window it is displayed in. If the SIZE is specified, the size of the form or window it is displayed in takes precedence. This means that the form will be "clipped" if the form size is larger than the size of the form or window it is displayed in. When the form size is smaller than the field size, the form size "grows" to fill the form or window field.

CREATE FORM Statement, ATTRIBUTE Clause

4.3.1.1.2 ATTRIBUTE Clause

This clause allows you to define your own options for clauses which call for an attribute_id in the definitions of any of the non-graphics fields that the form contains. If an attribute option is to be used as the form or window background, the ATTRIBUTE clause defining it must appear prior to the BACKGROUND clause (BACKGROUND is discussed in section 4.3.1.1.5 of this manual). The syntax for the ATTRIBUTE clause is:

[ATTRIBUTE attribute_id (Primitive [, ...])] ...

ATTRIBUTE is the clause keyword.

attribute_id is a unique name of up to 10 letters, numbers, and/or underscores. The attribute is specified to the Form Processor and Form Editor by this name. An attribute_id is required and cannot begin with a number. A list of pre-defined attribute_ids is contained in section 4.3.10 of this manual.

Primitive defines what the attribute means. One or more Primitive values may be specified. A list of valid primitives is contained in section 4.3.8 of this manual.

CREATE FORM Statement, KEYPAD Clause

4.3.1.1.3 KEYPAD Clause

This clause allows you to give a name to one of the terminal programmable function keys. This clause is optional. The syntax for the KEYPAD clause is:

KEYPAD ({ function_id = int } ...)

KEYPAD is the clause keyword.

function_name is the name you give to the function key you are defining. The name can be a maximum of 10 alphabetic characters. You will use this name in ON PICK condition statements to specify the key that determines the action.

int is the number of the programmable function key you wish to define. int can be any integer in the set {0, 2, 4 through 20, 105 through 120}.

CREATE FORM Statement, BACKGROUND Clause

4.3.1.1.4 BACKGROUND Clause

This clause allows you to define the background of the form. This is analogous to specifying what color of paper a paper form is printed on. This clause is optional. If it is omitted, the background of the form defaults to XPARNT. This means that the form inherits the background attribute of its containing form or window. The syntax for the BACKGROUND clause is:

BACKGROUND attribute_id

BACKGROUND is the clause keyword.

attribute_id is the name of the background attribute. It can be one of the pre-defined attribute_ids or one that you define in an ATTRIBUTE clause prior to this BACKGROUND clause definition.

CREATE FORM Statement, PROMPT Clause

4.3.1.1.5 PROMPT Clause

This clause allows you to specify the background text that will appear on the form. Prompts are optional and may be repeated as many times as needed to specify the background text for the form. The syntax for the PROMPT clause is:

[PROMPT Location prompt_string] ...

PROMPT is the clause keyword.

Location describes where the background text will be positioned on the form. The syntax for this parameter is described in section 4.3.4 of this manual.

prompt_string is the background text to be displayed on the form and must be enclosed in double quotes ("text").

The PROMPT clause may be repeated if more than one prompt is to be associated with the form being created.

Item Field Definition

4.3.1.2 Item Field Definition

An item field is a place-holder for one specific piece of data. The application program may read from and write to these fields. An item field may be non-display, that is, it exists, it can be referenced, it can contain data, but it will not be visible on the form. The syntax for an item field definition is:

```
ITEM item_id [ ( Repeat_Spec [ , ... ] ) ]  
[ Location ]  
[ SIZE int-1 [ BY int-2 ] ]  
[ VALUE Expression ]  
[ NODUP ]  
[ DISPLAY AS attribute_id ]  
[ DOMAIN ( Option ... ) ]  
{ HELP { string } form_id }  
{ APPLICATION }  
[ PROMPT Location prompt_string ... ]  
[ APPEARS IF Expression ]
```

ITEM Statement

4.3.1.2.1 ITEM Statement

This statement specifies that the form contains a data field. The syntax for the ITEM statement is:

ITEM item_id [(Repeat_Spec [, ...])]

ITEM is the clause keyword.

item_id is a unique name of up to 10 letters, numbers, and/or underscores. An application program can read from or write to an item field by providing the Form Processor with the qualified name of the item to be accessed. (See section 4.4 for further explanation of qualified names.) An item_id is required and cannot begin with a number.

Repeat_Spec specifies that the field appears on the form more than once and whether or not the area can be scrolled. A field may repeat, either horizontally or vertically, n times with m spaces between repetitions to form rows or columns. That repeat specification may then be repeated to form two dimensional arrays of fields. When an array needs to be scrolled, the number of actual data occurrences and how many occurrences should be displayed at one time are both specified. The syntax for this parameter is described in section 4.3.4 of this manual.

ITEM Statement, Location Clause

4.3.1.2.2 Location Clause

The Location clause specifies where a field or the area occupied by an array of fields will be positioned on the containing form. If the field is an open-ended array, the location is the position of the first occurrence of the field. If the Location clause is omitted, the field will not be displayed on the form. It will still be available for reference and you may store data in the field.

The syntax for Location is described in section 4.3.4 of this manual.

ITEM Statement, SIZE Clause

4.3.1.2.3 SIZE Clause

The SIZE clause determines the area on the form that each occurrence of the item occupies. This is specified by the width and height. An item may not overlap other fields or text and must be within the boundaries of its containing form. If a VALUE clause is included in the field definition, SIZE is optional for item fields. In this case, the width of the item defaults to the length of the VALUE string and the height is one. The syntax for the SIZE clause is:

SIZE int-1 [BY int-2]

SIZE is the clause keyword.

int-1 is the width of the field. It is expressed as the number of columns it occupies on the form.

BY is a reserved word that must be included when int-2 is specified.

int-2 is the height of the field. It is expressed as the number of rows it occupies on the form. This parameter is optional and defaults to one if not entered.

ITEM Statement, VALUE Clause

4.3.1.2.4 VALUE Clause

The VALUE clause allows you to specify the value for the item field as an expression. The expression is only evaluated when one of its components changes. If the field is enterable, the user can change the value. This clause is optional. If omitted, the item is blank filled. If a string is used to specify a default value and SIZE is omitted from the item definition, the item is assumed to be one dimensional and its length is the number of characters needed to display the string. The syntax for the VALUE clause is:

 VALUE Expression

 VALUE is the clause keyword.

 Expression specifies the value of the field. Valid expression types are explained in section 4.3.7 of this manual.

ITEM Statement, NODUP Clause

4.3.1.2.5 NODUP Clause

The NODUP clause is part of an item field on a form, like the size and location clauses. The item with the NODUP clause is the target of a SELECT action (see section 4.4.1.6.4 of the Application Generator User's Manual). In a sequence of rows selected from a database, if the value of a column in one row is the same as that of the previous row, the value of the item will be blanks. The use of the NODUP clause assumes that the rows have been ordered by the column which targets the item. The syntax for the NODUP clause is:

NODUP

NODUP is the clause keyword.

ITEM Statement, DISPLAY AS Clause

4.3.1.2.6 DISPLAY AS Clause

The DISPLAY AS clause controls access to the field. This clause is required for every item field definition that has a location (a non-display has no location). The syntax for the DISPLAY AS clause is:

DISPLAY AS attribute_id

DISPLAY AS is the clause keyword.

attribute_id is the name of the DISPLAY AS attribute. This can be one of the pre-defined attributes (refer to Section 4.3.10 for a list of pre-defined attributes) or you may define your own DISPLAY AS attribute using the ATTRIBUTE clause in the form definition for the containing form.

ITEM Statement, DOMAIN Clause

4.3.1.2.7 DOMAIN Clause

At this time, the DOMAIN clause allows you to reformat the value a user enters and specify entry requirements and restrictions for the item field. This clause is optional. When used, the options may be entered in any order. The options may be used in combination with one another. Care must be taken not to specify mutually exclusive options. These are noted in the explanations below. The syntax for the DOMAIN clause is:

DOMAIN (Option ...)

DOMAIN is the clause keyword.

option is the name of a pre-defined DOMAIN option. Valid options are as follows:

LEFT positions an entered value in the leftmost position of the field. Any leading blanks are removed. The result affects both the internal representation of the value and how it appears when displayed.

RIGHT positions an entered value so that it ends in the rightmost position of the field. The value is padded with leading blanks. The result affects both the internal representation of the value and how it appears when displayed.

If neither LEFT nor RIGHT is specified, the position of the data value will be as it was entered in the field. LEFT and RIGHT are mutually exclusive.

ITEM Statement, DOMAIN Clause

UPPER	converts all lower-case characters of a data value to upper-case.
LOWER	converts all upper-case characters of a data value to lower-case.
	If neither UPPER nor LOWER is specified, the data value remains as it was entered in the field. UPPER and LOWER are mutually exclusive.
NUMERIC	specifies that only integers will be accepted for the field value. If this option is not specified, all alphanumeric characters are accepted.
REAL	specifies that a real number may be entered into the field.
<u>MAXIMUM</u> real-1	specifies that only numeric values will be accepted for the field and that the highest value is the real number "real-1". The abbreviation "MAX" may be used to specify this option.
<u>MINIMUM</u> real-2	specifies that only numeric values will be accepted for the field and that the lowest value is the real number "real-2". The abbreviation "MIN" may be used to specify this option.
	If either MAXIMUM or MINIMUM is specified, the field becomes REAL. If you specify both the MAXIMUM and MINIMUM options, you can define a range of acceptable field values.
MUST ENTER	means that the user is required to enter a value for the item before the form can be processed by the application program.
MUST FILL	means that if the user enters a value for the item it must fill every position of the item. This option is typically used for items such as phone numbers or social security numbers.

ITEM Statement, DOMAIN Clause

PICTURE

the PICTURE option can be used to specify the format of items which are the targets of SELECT statements. To do this, specify

PICTURE (" format ")

where format is any valid COBOL picture clause. For example, "99.99" will specify a two digit numeric with two decimal places. DO NOT abbreviate. For example, do not use "x(3)". Instead use "xxx". DO NOT specify any numeric field as comp, comp-1, etc.

ITEM Statement, HELP Clause

4.3.1.2.8 HELP Clause

The HELP clause specifies a string or another form that is displayed when the cursor is in this field and the <HELP> key is pressed. It can also specify that the application program decides what to do next. The syntax for the HELP clause is:

```
HELP { string      }
      { form_id     }
      { APPLICATION }
```

HELP is the clause keyword.

string is a string of up to 60 characters enclosed in double quotes ("help_string") that is displayed in the message line of the screen when the <HELP> key is pressed.

form_id identifies another form that is displayed when the <HELP> key is pressed. Display of the form containing the field with the associated HELP clause is suppressed until the <ENTER> key is pressed. The help form can be defined in the current FDL source file or a different one.

APPLICATION specifies that the application program will decide what to do when the <HELP> key is pressed. The abbreviation AP may be used for this option.

ITEM Statement, PROMPT Clause

4.3.1.2.9 PROMPT Clause

The PROMPT clause allows you to specify information associated with the field such as labels and instructions. The syntax for the PROMPT clause is:

[PROMPT Location prompt_string] ...

PROMPT is the clause keyword.

Location specifies where the prompt appears when the form is displayed. The syntax for this parameter is described in section 4.3.4. Location is required. Care must be taken to ensure that the prompt location does not conflict with the item field location.

prompt_string is the information to be displayed enclosed in double quotes ("text").

The PROMPT clause may be repeated if more than one prompt is to be associated with the item field being created.

ITEM Statement, APPEARS IF Clause

4.3.1.2.10 APPEARS IF Clause

The APPEARS IF clause is used to specify when the item field appears on its containing form. If the Expression parameter evaluates as true, the field appears. If it evaluates to false, the field does not appear. If the APPEARS IF clause is omitted, the field always appears on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

APPEARS IF are the clause keywords.

Expression is the truth-valued criterion that determines when a field appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

Form Field Definition

4.3.1.3 Form Field Definition

A form field is a place-holder for a group of logically related data. Form fields are used to incorporate a subform within a form. The syntax for a form field definition is:

```
FORM form_id [ ( Repeat_Spec [ , ... ] ) ]  
    Location  
    SIZE int-1 [ BY int-2 ]  
    [ BACKGROUND attribute_id ]  
    [ PROMPT Location prompt_string ]  
    [ APPEARS IF Expression ]
```

FORM Statement

4.3.1.3.1 FORM Statement

This statement specifies that you are incorporating a subform into the form being defined. The subform must be defined with its own form definition. The form definition for the subform can be in the current FDL source file or a different one. The syntax for the FORM statement is:

FORM form_id [(Repeat_Spec [, ...])]

FORM is the statement keyword.

form_id is a unique name of up to 10 letters and/or numbers. This form id corresponds to the form_id on the CREATE FORM statement in the definition of the subform.

Repeat_Spec specifies that the subform appears on the form more than once and whether or not the area can be scrolled. A subform may repeat, either horizontally or vertically, n times with m spaces between repetitions to form rows or columns. That repeat specification may then be repeated to form two dimensional arrays of subforms. When an array needs to be scrolled, the number of actual data occurrences and how many occurrences should be displayed at one time are both specified. The syntax for this parameter is described in section 4.3.4 of this manual.

FORM Statement, Location Clause

4.3.1.3.2 Location Clause

This clause specifies where the field or the area occupied by an array of fields will be positioned on the containing form. If the field is an open-ended array, the location is the position of the first occurrence of the field. Repeating occurrences are then positioned relative to this occurrence based on the repeat specification. The syntax for this clause is described in section 4.3.4 of this manual.

FORM Statement, SIZE Clause

4.3.1.3.3 SIZE Clause

The SIZE clause determines the area on the form that each occurrence of the subform may occupy. The top left character of the form field becomes the origin of the subform it contains. The syntax for the SIZE clause is:

SIZE int-1 [BY int-2]

SIZE is the clause keyword.

int-1 is the width of the field. It is expressed as the number of columns it occupies on the form.

BY is a reserved word that must be included when int-2 is specified.

int-2 is the height of the field. It is expressed as the number of rows it occupies on the form. This parameter is optional and defaults to one if not entered.

The width and/or the height values may be specified as "*" to indicate that the size may change at run-time to accommodate changes in the size and location of fields on the form. If the size of a subform is not the same as the form field, the size of the form field takes precedence.

FORM Statement, PROMPT Clause

4.3.1.3.4 PROMPT Clause

The PROMPT clause allows you to display information associated with the field such as labels and instructions. The syntax for the PROMPT clause is:

[PROMPT Location prompt_string] ...

PROMPT is the clause keyword.

Location specifies where to position the information on the form containing the field. The syntax for this parameter is described in section 4.3.4.

prompt_string is the information to be displayed. It must be enclosed in double quotes ("text").

The PROMPT clause may be repeated if more than one prompt is to be associated with the form field being created.

FORM Statement, APPEARS IF Clause

4.3.1.3.5 APPEARS IF Clause

The APPEARS IF clause is used to specify when the form field appears on its containing form. If the Expression parameter evaluates to true, the field appears. If it evaluates to false, the field does not appear. If the APPEARS IF clause is omitted, the field always appears on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

Expression is the truth-valued criterion that determines when a field appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

Window Field Definition

4.3.1.4 Window Field Definition

Window fields are used as place-holders on a form for subforms to be supplied by the application program at run-time. The syntax for a window field definition is:

```
WINDOW window_id ( Repeat_Spec [ , ... ] )  
    Location  
    SIZE int-1 [ BY int-2 ]  
    BACKGROUND attribute_id  
    [ PROMPT Location prompt_string ] ...  
    [ APPEARS IF Expression ]
```

WINDOW Statement

4.3.1.4.1 WINDOW Statement

This statement specifies that you are defining a field on the form as a window. Its contents will be determined by the application at run-time. The syntax for the WINDOW statement is:

WINDOW window_id (Repeat_Spec [, ...])

WINDOW is the statement keyword.

window_id is a unique name of up to 10 letters, numbers and/or underscores. This name cannot begin with a number and is used by the application to tell the Form Processor where to put subforms.

Repeat_Spec specifies that the window appears on the form more than once and whether or not the area can be scrolled. A window may repeat, either horizontally or vertically, n times with m spaces between repetitions to form rows or columns. That repeat specification may then be repeated to form two dimensional arrays of fields. When an array needs to be scrolled, the number of actual data occurrences and how many occurrences should be displayed at one time are both specified. The syntax for this parameter is described in section 4.3.4 of this manual.

WINDOW Statement, Location Clause

4.3.1.4.2 Location Clause

This clause specifies where the field or the area occupied by an array of fields will be positioned on the containing form. If the field is an open-ended array, the location is the position of the first occurrence of the field. Repeating occurrences are then positioned relative to this occurrence based on the repeat specification. The syntax for this clause is described in section 4.3.4 of this manual.

WINDOW Statement, SIZE Clause

4.3.1.4.3 SIZE Clause

The SIZE clause determines the area on the form that each occurrence of the window may occupy. The top left character of the window field becomes the origin of any subforms it contains. The syntax for the SIZE clause is:

SIZE int-1 [BY int-2]

SIZE is the clause keyword.

int-1 is the width of the field. It is expressed as the number of columns it occupies on the form.

BY is a reserved word that must be included if int-2 is specified.

int-2 is the height of the field. It is expressed as the number of rows it occupies on the form. This parameter is optional and defaults to one if not entered.

When a subform is displayed in a fixed size window, if the form size is bigger than the reserved window, the form is "clipped". If the form is smaller, it will "grow" to fill the window.

WINDOW Statement, BACKGROUND Clause

4.3.1.4.4 BACKGROUND Clause

This clause allows you to define the background of the window field. This is analogous to the print and paper colors of a paper form. This clause is required. The syntax for the BACKGROUND clause is:

BACKGROUND attribute_id

BACKGROUND is the clause keyword.

attribute_id is the name of the background attribute. It can be one of the pre-defined attribute_ids or one that you define in an ATTRIBUTE clause prior to the BACKGROUND clause for the containing form.

WINDOW Statement, PROMPT Clause

4.3.1.4.5 PROMPT Clause

The PROMPT clause allows you to specify information associated with the field such as labels and instructions. The syntax for the PROMPT clause is:

[PROMPT Location prompt_string] ...

PROMPT is the clause keyword.

Location specifies where to position the information on the form containing the field. The syntax for this parameter is described in section 4.3.4 in this manual.

prompt_string is the information to be displayed. It must be enclosed in double quotes ("text").

The PROMPT clause may be repeated if more than one prompt is to be associated with the window field being created.

WINDOW Statement, APPEARS IF Clause

4.3.1.4.6 APPEARS IF Clause

The APPEARS IF clause is used to specify when the window field appears on its containing form. If the Expression parameter evaluates to true, the field appears. If it evaluates to false, the field does not appear. If the APPEARS IF clause is omitted, the field always appears on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

Expression is the truth-valued criterion that determines when a field appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

Graph Field Definition

4.3.1.5 Graph Field Definition

A graph can be defined as a type of field on a form, just like an item, a form, or a window. A graph field is similar to a form field, in that the graph field reserves an area on the form into which a graph defined elsewhere is drawn at run-time, just like a form field reserves an area into which is drawn a form which is defined elsewhere. Like an item field, a graph field is a terminus on the display list, that is, it cannot contain other fields or forms like a form field or window field can. The syntax for defining a graph field on a form is:

```
GRAPH graph_name
      Location
      SIZE int-1 BY int-2
      [ PROMPT location prompt_string ] ...
      [ APPEARS IF Expression ]
```

GRAPH Statement

4.3.1.5.1 GRAPH Statement

The GRAPH statement specifies that you are defining a field on a form as a graph field. Its contents will be determined by the definition of the GRAPH which is specified by graph_name. The syntax for the GRAPH statement is:

GRAPH graph_name

GRAPH is the statement keyword.

graph_name is the name of the graph (defined elsewhere) which is to be drawn into the graph field at run-time.

GRAPH Statement, Location Clause

4.3.1.5.2 Location Clause

The Location clause specifies where the upper left-hand corner of the graph field will be positioned on the containing form. The syntax for Location is defined in section 4.3.4 of this manual.

GRAPH Statement, SIZE Clause

4.3.1.5.3 SIZE Clause

The SIZE clause determines the area on the form that each occurrence of the graph field occupies. This is specified by width and height. A graph field may not overlap other fields or text and must be within the boundaries of its containing form. The syntax for the SIZE clause is:

SIZE int-1 BY int-2

SIZE is the clause keyword.

int-1 is the width of the field, expressed as the number of columns the field occupies on the form.

BY is a reserved word that must be included.

int-2 is the height of the field, expressed as the number of rows the field occupies on the form.

GRAPH Statement, PROMPT Clause

4.3.1.5.4 PROMPT Clause

The PROMPT clause allows you to specify information associated with the graph field such as labels and instructions. A PROMPT will be placed outside of the graph field, just as a prompt for an item field is placed outside of the item. Since the PROMPT exists outside of the graph field, the prompt is composed of non-graphics characters. The syntax for the PROMPT clause is:

[PROMPT Location prompt_string] ...

PROMPT is the clause keyword.

Location specifies where the information appears when the form containing the graph field is displayed. The syntax for this parameter is described in section 4.3.4.

prompt_string is the text to be displayed enclosed in double quotes ("text").

The PROMPT clause can be repeated if more than one prompt is desired for the graph being created.

GRAPH Statement, APPEARS IF Clause

4.3.1.5.5 APPEARS IF Clause

The APPEARS IF clause is used to specify when the graph field appears on its containing form. If the Expression parameter evaluates to true, the field appears. If it evaluates to false, the field does not appear. If the APPEARS IF clause is omitted, the field always appears on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

APPEARS IF are the clause keywords.

Expression is the truth-valued criterion that determines when a field appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

Graph Definition

4.3.2 Graph Definition

A graph can either be brought into a form through use of a graph field, much like a subform is brought in through a form field, or it may exist independently of a form. In either case, the graph must be defined using the following syntax:

```
CREATE { BAR } GRAPH graph_name
       ( PIE )
       ( LINE )

       [ Location ]

       [ SIZE int-1 BY int-2 ]

       [ USING ( [ independent data ] [ AXIS axis_name ] ) ]

       [ { LEGEND { HORIZONTAL } Location { BOX } }
          { VERTICAL } ]
       [ { LABEL { ( Location, [ DISPLAY AS attribute_id ] ) }
          { DISPLAY AS attribute_id, Location } }
          { string } ] ...

       [ ATTRIBUTE attribute_id { LINE
                                  { FILL
                                  { PROMPT
                                  { MARKER } } } } (Primitive ...) ] ...

       [ BACKGROUND attribute_id ]
```

Graph Definition

```
AXIS axis_name
  DISPLAY AS attribute_id
    { HORIZONTAL }
    { VERTICAL }
    { }

  [ LABEL attribute_id Location string ] ...
  [ Location ]
  [ MINIMUM lower_limit ]
  [ MAXIMUM upper_limit ]
  [ SCALE { LINEAR }
    { LOG } ]
  [ SIZE length ]
    [ TICK { RIGHT
      { LEFT
        { ABOVE
          { BELOW } } } } ]
    [ [ EVERY ] ndiv ] [ minor ]
    [ attribute_id ] [ string ... ] ]
  [ [ FINE ] GRID ] ] ...

CURVE curve_name
  dependent data [ USING AXIS axis_name ]
    [ VERSUS independent data ]
  [ DISPLAY AS attribute_id ]
  [ LEGEND attribute_id string ]
  [ SHADE COLOR color ]
  [ ABSOLUTE
    { ADDITIVE USING CURVE curve_name } ] ] ...
```

Graph Definition

```
[ PIE int
  [ EXPLODE fraction ]
  [ LABEL attribute_id [ Location ] string ]
  [ LEGEND attribute_id string ]
  [ PERCENT { INSIDE } attribute_id ]
  [ PERCENT { OUTSIDE } attribute_id ]
  [ PERCENT { Location } attribute_id ]
  [ QUANTITY { INSIDE } attribute_id ]
  [ QUANTITY { OUTSIDE } attribute_id ]
  [ QUANTITY { Location } attribute_id ]
  [ SHADE COLOR color ]
]...  
..
```

CREATE GRAPH Statement

4.3.2.1 CREATE GRAPH Statement

Every graph must be defined with a CREATE GRAPH statement. This statement tells the compiler that what follows is a graph definition. It also signals the end of the previous form or graph definition if it is not the first one in the source file. The syntax for a CREATE GRAPH statement is:

```
( BAR )  
CREATE { PIE } GRAPH graph_name  
( LINE )
```

CREATE GRAPH are the statement keywords.

BAR
LINE
PIE

are keywords which specify which type of graph to create. You must choose one of these options.

graph_name is a unique name of up to 10 letters and/or numbers associated with each graph. It is used by the application program to identify the graph. A graph_name is required. A graph_name cannot begin with a number.

CREATE GRAPH Statement, Location Clause

4.3.2.2 Location Clause

The Location clause specifies where the upper left-hand corner of the graph field will be positioned in the containing window or form. The syntax for Location is defined in section 4.3.4 of this manual.

CREATE GRAPH Statement, SIZE Clause

4.3.2.3 SIZE Clause

The SIZE clause determines the area on the form that each occurrence of the graph field occupies. This is specified by width and height. A graph field may not overlap other fields or text and must be within the boundaries of its containing form. The syntax for the SIZE clause is:

SIZE int-1 BY int-2

SIZE is the clause keyword.

int-1 is the width of the field, expressed as the number of columns the field occupies on the form.

BY is a reserved word that must be included.

int-2 is the height of the field, expressed as the number of rows the field occupies on the form.

CREATE GRAPH Statement, USING Clause

4.3.2.4 USING Clause

The USING clause is used to specify where the graph is to draw the independent data from and which axis of the graph the independent data is to be associated with. The syntax for the USING clause is:

USING ([independent data] [AXIS axis_name])

USING is the clause keyword.

independent data is either a list of path names to items from which the independent data is to be drawn, or a list of constants which make up the independent data. The path names must specify numeric items or an array of numeric items. The constants must be numeric constants. If you have specified more curves than items, the "leftover" curves will default to 0. The items in the lists, whether paths or constants, must be separated by at least one space. You may separate the items with a comma (,) for clarity, but the comma is not required. The following examples are all valid entries for independent data:

1, 3, 5, 7

'formname.item1', 'formname.item2'

'formname.array'

'formname(*).item'

AXIS is a keyword to specify that what follows is an axis name.

axis_name is the name of the axis on which to plot the independent data. The axis name must be an alphanumeric name of not more than 10 characters. It must begin with an alpha character.

CREATE GRAPH Statement, LEGEND Clause

4.3.2.5 LEGEND Clause

The LEGEND clause is optional. It is used to specify the location and format of the legend to be associated with the graph, if a legend is desired. The syntax for the LEGEND clause is:

```
LEGEND { HORIZONTAL } { VERTICAL } Location [ BOX ]
```

LEGEND is the keyword to specify a LEGEND clause.

HORIZONTAL

VERTICAL

are keywords used to specify whether you want the legend to appear vertically on the screen or horizontally (e.g., L E G E N D or L
E
G
E
N
D).

You may abbreviate with either H or V. If neither HORIZONTAL nor VERTICAL is specified, the default is VERTICAL.

Location

specifies where the legend is to appear within the graph form. The syntax for location is described in section 4.3.4 of this manual.

BOX

is a keyword used to specify whether you want a box drawn around the legend. If the word BOX is used, a box will be drawn. If BOX is omitted, no box will be drawn.

CREATE GRAPH Statement, LABEL Clause

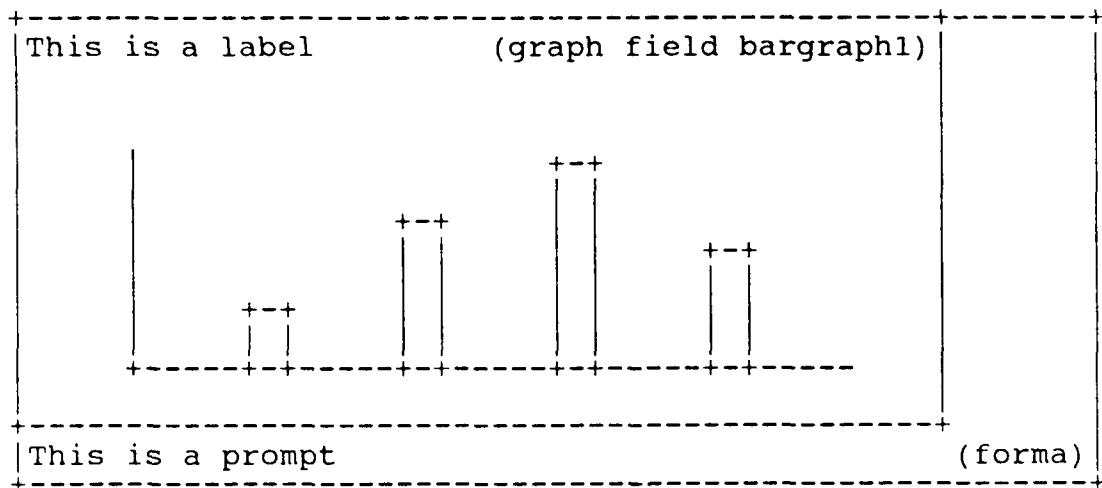
4.3.2.6 LABEL Clause

The LABEL clause is used to specify a label within the graph field. Since the LABEL exists within the graph field, it is composed of graphics characters. The PROMPT clause associated with the GRAPH field definition specifies a label which appears outside of the graph field. It is composed of non-graphics characters. For example, the following FDL code:

```
CREATE FORM forma
.
.
.
GRAPH bargraph1
  AT 1,1
  SIZE ...
  PROMPT AT 1 below "This is a prompt"

CREATE BAR GRAPH bargraph1
  USING ...
  LEGEND ...
  LABEL AT 1,1 "This is a label"
```

defines the following graph:



CREATE GRAPH Statement, LABEL Clause

The syntax for specifying a label is:

```
[ { Location, [ DISPLAY AS attribute_id ) ]  
  LABEL { DISPLAY AS attribute_id, Location  
         ( ) }  
         string } ] ...
```

LABEL is the clause keyword.

Location specifies where the label will be positioned in the graph field. If Location is specified first, DISPLAY AS is not required, although you may include it if desired. Location is always required.

DISPLAY AS are keywords which specify that an attribute which applies to the label will follow.

attribute_id is the name of the DISPLAY AS attribute. Currently, the only attributes which may be applied to labels are colors. Available colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

string is the string enclosed in double quotes (" ") which is to be used as the label.

If the DISPLAY AS Clause is not specified, an attribute will be chosen based on the background color of the graph field and previously chosen colors for the graphics element type.

The LABEL clause may be repeated, that is, a graph may have multiple labels.

CREATE GRAPH Statement, ATTRIBUTE Clause

4.3.2.7 ATTRIBUTE Clause

This clause allows you to define your own options for the BACKGROUND, DISPLAY AS, TICK, LEGEND, PERCENT, QUANTITY, and LABEL clauses in the definitions of any of the fields that the form contains. The syntax for defining an attribute is:

```
[ ATTRIBUTE attribute_id [ LINE  
                           FILL  
                           PROMPT  
                           MARKER ] ( Primitive ... ) ] ...
```

ATTRIBUTE is the clause keyword.

attribute_id is the name of the attribute being defined.

LINE

FILL

PROMPT

MARKER

are keywords used to specify the type of element with which the attribute will be associated.

Specify LINE if the attribute is to be associated with a polyline, MARKER if the attribute is to be associated with a polymarker, PROMPT if the attribute is to be associated with text, or FILL if the attribute is to be associated with a fillarea.

Primitive defines what the attribute means. Refer to section 4.3.9 for a list of valid primitives.

The ATTRIBUTE clause may be repeated, that is, you may define multiple attributes for a graph.

CREATE GRAPH Statement, BACKGROUND Clause

4.3.2.8 BACKGROUND Clause

This clause allows you to define the background of the graph. This is analogous to specifying what color paper the graph is printed on. This clause is optional. If it is omitted, the background of the graph defaults to XPARNT. This means that the graph inherits the background attribute of its containing form or window. The syntax for the BACKGROUND clause is:

BACKGROUND attribute_id

BACKGROUND is the clause keyword.

attribute_id is the name of the attribute to be assigned. The attribute_id can be either one which is pre-defined (refer to section 4.3.10 for a list of pre-defined attribute_ids), or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual.

CREATE GRAPH Statement, AXIS Clause

4.3.2.9 AXIS Clause

The AXIS clause applies to BAR and LINE graphs only. It is used to define the axes which delimit the graph. The AXIS clause includes several options which can enhance the definition of your graph. The syntax for the AXIS clause is:

```
AXIS axis_name
      DISPLAY AS attribute_id
      ( HORIZONTAL )
      ( VERTICAL )
      ( )
      [ LABEL attribute_id [ Location ] string ] ...
      [ Location ]
      [ MINIMUM lower_limit ]
      [ MAXIMUM upper_limit ]
      [ SIZE length ]
      [ TICK [ RIGHT ]
            [ LEFT ]
            [ ABOVE ]
            [ BELOW ]
            [ ]
            [ attribute_id ] [ string ... ] ]
      [ [ FINE ] GRID ]
```

AXIS is the clause keyword.

axis_name is the name of the axis being defined.

The various phrases associated with the AXIS clause are explained in the following sections.

AXIS Clause, DISPLAY AS Phrase

4.3.2.9.1 DISPLAY AS Phrase

The DISPLAY AS phrase is used to specify an attribute associated with the AXIS being defined. DISPLAY AS is required. The syntax for the DISPLAY AS phrase is:

DISPLAY AS attribute_id

DISPLAY AS are the phrase keywords.

attribute_id is the name of the attribute to be associated with the axis being defined. The attribute_id can be either one which is pre-defined (refer to section 4.3.10 for a list of pre-defined attribute_ids), or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual.

AXIS Clause, DIRECTION Phrase

4.3.2.9.2 DIRECTION Phrase

Direction is used to specify the orientation of the axis being defined. Direction is required. The syntax for the DIRECTION phrase is:

(HORIZONTAL)
{ VERTICAL }
()

HORIZONTAL specifies that the axis being defined is horizontal (can be abbreviated to H).

VERTICAL specifies that the axis being defined is vertical (can be abbreviated to V).

Either HORIZONTAL or VERTICAL may be specified, but not both.

AXIS Clause, LABEL Phrase

4.3.2.9.3 LABEL Phrase

The LABEL phrase is used to specify a label to be associated with the axis being defined. If the axis is horizontal the label will appear horizontally underneath the axis. If the axis is vertical, the label will appear vertically to the left of the axis. The syntax for the LABEL phrase is:

[LABEL attribute_id [Location] string] ...

LABEL is the phrase keyword.

attribute_id is the name of the attribute to be associated with the label for the axis. The attribute_id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3. A list of pre-defined attribute_ids is contained in section 4.3.10 of this manual.

Location specifies where the first character of the label string is to be placed. The syntax for Location is explained in section 4.3.4 of this manual.

string is the text to be used as the label enclosed in double quotes ("").

AXIS Clause, Location Phrase

4.3.2.9.4 Location Phrase

The location is the origin of the axis specified by row and column. The syntax for location is described in section 4.3.4 of this manual. The following diagram shows the origin of a pair of axes:

| x marks the origin of each of the axes.
| The vertical axis originates at x and
| is drawn upwards. The horizontal axis
| originates at x and is drawn to the
| right.

| x-----

AXIS Clause, MINIMUM Phrase

4.3.2.9.5 MINIMUM

MINIMUM is a keyword used to specify the minimum value of the data to be plotted on the graph. This specifier is optional. If MINIMUM is not specified, the graph will begin with the lowest data value which will provide "nice" values for the range specified for the axis. (A "nice" value is one that is some multiple of 1, 2, or 5. For instance, a range of 8 to 48 would produce an axis which begins with 5 and ends with 50.) If MINIMUM is specified, plotting will begin at the specified value. The syntax for the MINIMUM phrase is:

MINIMUM lower_limit

MINIMUM is the phrase keyword.

lower_limit is the lowest data value in your range.

AXIS Clause, MAXIMUM Phrase

4.3.2.9.6 MAXIMUM Phrase

MAXIMUM is a keyword used to specify the maximum value of the data to be plotted on the graph. This specifier is optional. If MAXIMUM is not specified, the graph will stop plotting with the highest data value which will produce "nice" values for the range specified for the axis. (A "nice" value is one that is some multiple of 1, 2, or 5. For instance, a range of 8 to 48 would produce an axis which begins with 5 and ends with 50.) If MAXIMUM is specified, plotting will end with the specified value. The syntax for the MAXIMUM phrase is:

MAXIMUM upper_limit

MAXIMUM is the phrase keyword.

upper_limit is the highest data value in your range.

AXIS Clause, SCALE Phrase

4.3.2.9.7 SCALE Phrase

The SCALE phrase specifies whether the scale used on the axis is to be linear or logarithmic. The syntax for the SCALE phrase is:

```
SCALE { LINEAR }
      { LOG      }
      (         )
```

SCALE is the phrase keyword.

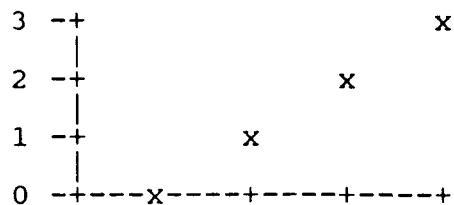
LINEAR is a keyword which specifies that the scale is to be linear. If SCALE is not specified, LINEAR is the default value.

LOG specifies that the scale of the axis is to be logarithmic. This means that the data to be plotted will be the exponent which produced the value rather than the actual value, resulting in a linear curve. For example, for the values 1 (100), 10 (101), and 100 (103), the linear curve will be:

```
100 --+ x
      90 --+
      80 --+
      70 --+
      60 --+
      50 --+
      40 --+
      30 --+
      20 --+
      10 --+ x
      1 --+ x
      0 -----+-----+-----+
```

AXIS Clause, SCALE Phrase

If, however, a logarithmic scale has been specified, the curve will be plotted against the exponents rather than the values, and the curve will look like this:



resulting in a graph which is much easier to read and interpret.

AXIS Clause, SIZE Phrase

4.3.2.9.8 SIZE Phrase

The SIZE phrase is used to specify the length (in character cells) of the axis being defined. A horizontal axis is defined by character cell widths; a vertical axis is defined in character cell heights. For example, a horizontal axis defined with a length of 10 will be 10 characters wide. A vertical axis defined with a length of 10 will be 10 characters high. (NOTE: on most terminals, a character cell is approximately twice as high as it is wide. This means that a horizontal axis with a length of 10 will be about half as long as a vertical axis with a length of 10. This means that if you switch a vertical axis to a horizontal one, or vice versa, you must remember to resize it accordingly.) The syntax for the SIZE phrase is:

SIZE length

SIZE is the phrase keyword.

length is the length of the axis specified in character cells.

AXIS Clause, TICK Phrase

4.3.2.9.9 TICK Phrase

The TICK phrase is used to specify TICK marks on an axis if they are desired. You may specify major tick marks, minor tick marks, the number of marks, and the spacing of the marks. The syntax for the TICK phrase is:

```
[ TICK { RIGHT
          LEFT
          ABOVE
          BELOW
          [ attribute_id ] [ string ... ] ]
          [ [ EVERY ] ndiv ] [ minor ] ]
```

TICK is the phrase keyword.

RIGHT is a keyword which specifies that the tick marks are to appear to the right of the vertical axis, or below the horizontal axis, depending upon which axis the RIGHT keyword is associated with. For example, RIGHT would be used to produce the two axes shown below:

+-+-+-+---+-+---+-+---+-+---+-+
+-+-+-+---+-+---+-+---+-+---+-+
+-+-+-+---+-+---+-+---+-+---+-+
+-+-+-+---+-+---+-+---+-+---+-+
+-+-+-+---+-+---+-+---+-+---+-+
+-+-+-+---+-+---+-+---+-+---+-+

LEFT is a keyword which specifies that the tick marks are to appear to the left of the vertical axis, or above the horizontal axis, depending upon which axis the LEFT keyword is associated with. For example, LEFT would be used to produce the two axes shown below:

AXIS Clause, TICK Phrase

BELOW is a keyword which specifies that the tick marks are to appear below the horizontal axis, or to the right of the horizontal axis, depending upon which axis the BELOW keyword is associated with. For example, BELOW would be used to produce the two axes shown below:

```
+-+-+---+---+---+  
+-  
+-  
+-  
+-  
+-  
+-
```

ABOVE is a keyword which specifies that the tick marks are to appear above the horizontal axis, or to the left of the vertical axis, depending upon which axis the ABOVE keyword is associated with. For example, ABOVE would be used to produce the two axes shown below:

```
+-+-+---+---+---+  
-+  
--  
--  
--  
--  
--
```

EVERY is a keyword used to specify whether the following number (ndiv) refers to the number of major tick marks or to the number of steps between each major tick mark.

ndiv without the keyword EVERY preceding it, ndiv specifies the number of major tick marks. If EVERY is used, ndiv specifies the number of steps between each major tick mark.

For example, if TICK 5 is used in the definition of a horizontal axis with data values of 0, 20, and 40, the following is produced:

```
-----+  
0 10 20 30 40
```

AXIS Clause, TICK Phrase

If TICK EVERY 5 is used on a horizontal axis and the data values to be plotted are 0, 20, and 40, the following is produced:

```
+---+---+---+---+---+---+---+  
0 5 10 15 20 25 30 35 40
```

minor is the number of minor tick marks to place between the major tick marks, if minor ticks are desired. For example, TICK 5 2 specifies an axis with 5 major tick marks and 2 minor tick marks between the major tick marks, as follows:

```
+---+---+---+---+---+---+  
| | | | | | |
```

attribute_id the name of the attribute to apply to the tick mark label(s). The attribute_id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual. A list of pre-defined attribute ids is contained in section 4.3.10 of this manual.

string the label(s), enclosed in double quotes (" ") to associate with each tick mark. If you do not supply labels, numeric labels will be supplied for you. For example, the following syntax for a horizontal axis: TICK EVERY 5 2 with data values of 0, 20, and 40 and no labels specified will produce the following:

```
+-----+-----+-----+-----+  
| | | | |  
0 5 10 15 20 25 30 35 40
```

If you specify the labels for the tick marks, the syntax looks like this: TICK EVERY 5 2 "85" "86" "87" "88" "89" "90" "91" "92" "93" and the axis looks like this:

```
+-----+-----+-----+-----+  
| | | | |  
85 86 87 88 89 90 91 92 93
```

AXIS Clause, TICK Phrase

If you specify fewer labels than you have specified tick marks, the labels repeat. For example, if you specify TICK EVERY 5 2 "85" "86" "87" "88" for the same set of data (0, 20, and 40), the axis produced is:

+++++
| 85 86 87 88 85 86 87 88 85

AXIS Clause, GRID Phrase

4.3.2.9.10 GRID Phrase

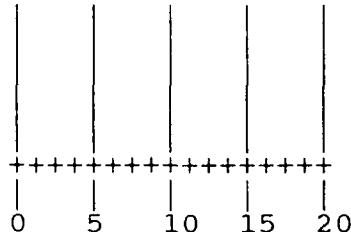
The GRID phrase is used to specify that you want a grid connected to your tick marks. It is not necessary to specify the TICK phrase in order to specify GRID. If TICK is not specified, a grid connected to the generated tick marks will be provided. The syntax for the GRID phrase is:

[FINE] GRID

FINE is a keyword used to specify that you want a fine grid connected to your minor tick marks.

GRID is the phrase keyword.

If you specify GRID, a grid will be provided running from your major tick marks, as follows:



If you specify FINE GRID, a grid will be provided running from your major tick marks and a finer set of lines will be provided running from the minor tick marks.

CREATE GRAPH Statement, CURVE Clause

4.3.2.10 CURVE Clause

The CURVE clause is used when defining BAR and LINE graphs. One CURVE clause is required for each line or set of bars being defined. The syntax for a CURVE clause is:

```
[ CURVE curve_name
    dependent data [ USING AXIS axis_name]
                    [ VERSUS independent data ]
    [ DISPLAY AS attribute_id ]
    [ LEGEND attribute_id string ]
    [ SHADE COLOR color ]
    [ ABSOLUTE
      ADDITIVE USING CURVE curve_name ] ] ...
```

CURVE is the clause keyword.

curve_name is the name of the curve being defined.

dependent data can be either a list of path names to items from which the dependent data is to be drawn, or a list of constants which make up the dependent data. The path names must specify numeric items or an array of numeric items. The constants must be numeric constants. If you have specified more curves than items, the "leftover" curves will default to 0. The items in the lists, whether paths or constants, must be separated by at least one space. You may separate the items with a comma (,) for clarity, but the comma is not required. The following examples are all valid entries for dependent data:

```
1, 3, 5, :  
'formname.item1', 'formname.item2'  
'formname.array'
```

The phrases associated with the CURVE clause are explained in detail in the following sections.

CURVE Clause, USING Phrase

4.3.2.10.1 USING Phrase

The USING phrase is used to specify which axis to associate the dependent data with. USING is optional. If you do not specify an axis with USING, the dependent data will be associated with whichever axis is NOT associated with the independent data. The syntax for the USING phrase is:

USING AXIS axis_name

USING is the phrase keyword.

AXIS is a required keyword, included for clarity.

axis_name is the name of the axis with which to associate the dependent data.

CURVE Clause, VERSUS Phrase

4.3.2.10.2 VERSUS Phrase

The VERSUS phrase is used to signify that the curve is to be graphed against an alternate set of independent data than that specified in the USING clause associated with the CREATE GRAPH statement. VERSUS is optional. The syntax for the VERSUS phrase is:

VERSUS independent data

VERSUS is the phrase keyword.

independent data specifies the lists of alternate independent data against which to graph the dependent data. The alternate independent data is specified in the same manner as the dependent data and the primary independent data, with either a list of path names or a list of numeric constants.

CURVE Clause, DISPLAY AS Phrase

4.3.2.10.3 DISPLAY AS Phrase

The DISPLAY AS Phrase is used to specify an attribute to be associated with either the line itself in a line graph, or the line which outlines the bar in a bar graph. DISPLAY AS is optional. If you do not specify a DISPLAY AS attribute, a color will be chosen based upon the background color and the last color chosen for that element type. The syntax for the DISPLAY AS phrase is:

DISPLAY AS attribute_id

DISPLAY AS are the phrase keywords.

attribute_id is the name of the attribute to associate with the line. The attribute_id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual. A list of pre-defined attribute_ids is contained in section 4.3.10 of this manual.

CURVE Clause, LEGEND Phrase

4.3.2.10.4 LEGEND Phrase

The LEGEND phrase is used to specify a text string to be associated with the legend for the curve being defined. LEGEND is optional. If LEGEND is not specified, the name of the curve will be used to identify the legend. The syntax for the LEGEND phrase is:

LEGEND attribute_id string

LEGEND is the phrase keyword.

attribute_id is the name of the attribute to associate with the string specified for the legend. The attribute_id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual. A list of pre-defined attribute_ids is contained in section 4.3.10 of this manual.

string is the text enclosed in double quotes (" ") to associate with the legend for the curve being defined.

CURVE Clause, SHADE COLOR Phrase

4.3.2.10.5 SHADE COLOR Phrase

The SHADE COLOR phrase is used to specify that the area delineated by the curve is to be filled in with a color. For a bar graph, the area inside of the bar is to be filled. For a line graph, the area beneath the line is to be filled. SHADE COLOR is optional. The syntax for SHADE COLOR is:

SHADE COLOR color

SHADE COLOR are the phrase keywords.

color is the color with which to fill in the bar or fillarea under a line. Available colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

CURVE Clause, ABSOLUTE Phrase

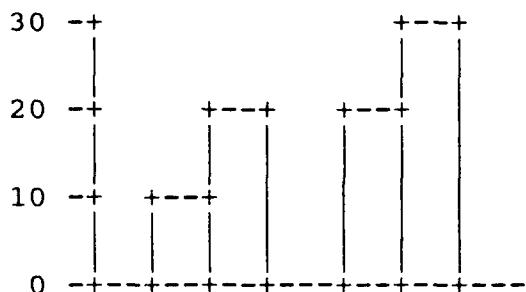
4.3.2.10.6 ABSOLUTE Phrase

ABSOLUTE specifies that the curve is to be a stand-alone curve. Each data value is to be represented by a separate bar or line. This means that for a bar graph, the bars will appear next to each other. For a line graph, depending on the specific data value, the lines may cross each other.

The following graph is absolute:

```
CURVE curve1
  "10, 20"
  .
  .
  .
  ABSOLUTE
```

```
CURVE curve2
  "20, 30"
  .
  .
  .
  ABSOLUTE
```



The syntax for the ABSOLUTE phrase is:

ABSOLUTE

ABSOLUTE is the phrase keyword.

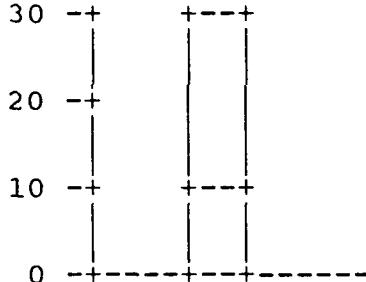
CURVE Clause, ADDITIVE Phrase

4.3.2.10.7 ADDITIVE Phrase

ADDITIVE specifies that the data value graphed by the curve or line is to be added to the specified curve before presenting on the screen. This means that for bar and line graphs, the bars or lines will appear to be stacked. ABSOLUTE and ADDITIVE are mutually exclusive. An example of an additive bar graph follows:

```
CURVE curve1
  "10"
  .
  .
  .
  ABSOLUTE

CURVE curve2
  "20"
  .
  .
  .
  ADDITIVE USING CURVE curve1
```



The syntax for the ADDITIVE phrase is:

ADDITIVE USING CURVE curve_name

ADDITIVE is the phrase keyword.

USING CURVE are keywords to specify the name of the curve to which the curve being defined is to be added.

curve_name is the name of the curve to which the curve being defined is to be added.

PIE Clause

4.3.2.11 PIE Clause

The PIE clause is used to specify segments of pie graphs. A PIE clause has no meaning for either bar or line graphs. The PIE clause is not required when defining a pie graph. If the PIE clause is not included, the pie chart will have the following characteristics: The segments will be given colors based upon the background color and the last color chosen for a segment, the segment labels will be percentages shown outside of the segments, there will be no labels for the legend, and none of the segments will be exploded out from the graph. The syntax for the PIE clause follows:

```
PIE int
      EXPLODE fraction ]
      LABEL attribute_id [ Location ] string ]
      LEGEND attribute_id string ]
      [ PERCENT { INSIDE } attribute_id ]
      [ { OUTSIDE } attribute_id ]
      [ ( Location ) ]
      [ QUANTITY { INSIDE } attribute_id ]
      [ { OUTSIDE } attribute_id ]
      [ ( Location ) ]
      [ SHADE COLOR color ]
PIE           is the clause keyword.
int          a number that specifies which occurrence of data
              the pie segment being defined is to represent.
              For instance, if you define pie segment 3, and
              the data set being represented by the pie graph
              is {4, 7, 9, 46}, then the third occurrence of
              data (9) will be represented by that segment.
              If the data set being represented is {34, 76},
              then there will be no third segment on the pie
              chart.
```

The various phrases associated with the PIE clause are described in detail in the following sections.

PIE Clause, EXPLODE Phrase

4.3.2.11.1 EXPLODE Phrase

The EXPLODE phrase is used when you wish to offset one or more segments from the graph. The syntax for the EXPLODE phrase is:

EXPLODE fraction

EXPLODE is the phrase keyword.

fraction is the percentage of the radius of the graph by which you wish to offset the segment. For example, if the radius of the pie graph is 10 and a segment was defined with an explosion factor of 10, then the segment will be offset from the graph by $1 * \text{whatever unit of measure the radius for that specific graph is}$ drawn in. (The size of the graph is determined by the size of the area it is to be drawn in, the location of the graph, the placement of the legend, labels, and so on. Thus, depending upon the options specified, the actual size of the graph might vary.

PIE Clause, LABEL Phrase

4.3.2.11.2 LABEL Phrase

The LABEL phrase is used to specify a label associated with the segment which is to be placed outside of the segment. The LABEL phrase is optional. The syntax for the LABEL phrase is:

LABEL attribute_id [Location] string

LABEL is the phrase keyword.

attribute_id is an attribute to be associated with the label. The attribute_id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual. A list of pre-defined attribute_ids is contained in section 4.3.10 of this manual.

Location specifies where the first character of the label is to be located. The syntax for location is listed in section 4.3.4 of this manual.

string is a string of text within double quotes (" ") which will appear outside of the segment.

PIE Clause, LEGEND Phrase

4.3.2.11.3 LEGEND Phrase

The LEGEND phrase is used to specify a string to be associated with the legend for the segment being defined. The LEGEND phrase is optional. The syntax for the LEGEND phrase is:

LEGEND attribute_id string

LEGEND is the phrase keyword.

attribute_id is the attribute to be associated with the string which labels the legend. The attribute id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual. A list of pre-defined attribute ids is contained in section 4.3.10 of this manual.

string is a string of text within double quotes (" ") which will be associated with the legend for the segment being defined.

PIE Clause, PERCENT Phrase

4.3.2.11.4 PERCENT Phrase

The PERCENT phrase is used to specify that a label is to be associated with the segment. The label will be a percentage based upon the data value being represented by the segment and the total value being represented by the entire graph. PERCENT is optional. If neither PERCENT nor QUANTITY nor LABEL is specified, a percent label will be supplied outside of the segment. The syntax for the PERCENT phrase is:

```
PERCENT { INSIDE
          { OUTSIDE } attribute_id
          { Location }
```

PERCENT is the phrase keyword.

INSIDE
OUTSIDE are keywords used to specify whether you want the percentage label to be placed inside of the segment or outside of it. If the label is to be placed inside of the segment, make sure any colors specified by attribute_id contrast with the color of the segment.

Location specifies a specific location where you want the percent label to be located, rather than having the location inside or outside chosen for you. The syntax for Location is listed in section 4.3.4 of this manual.

attribute_id is an attribute to be associated with the text of the percentage label. The attribute_id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual. A list of pre-defined attribute_ids is contained in section 4.3.10 of this manual.

PIE Clause, QUANTITY Phrase

4.3.2.11.5 QUANTITY Phrase

The QUANTITY phrase is used to specify that a label is to be associated with the segment. The label will be the actual data value represented by the segment. QUANTITY is optional. If neither PERCENT nor QUANTITY nor LABEL is specified, a percent label will be supplied outside of the segment. The syntax for the QUANTITY phrase is:

```
QUANTITY { INSIDE } attribute_id
          { OUTSIDE }
          ( Location )
```

QUANTITY is the phrase keyword.

INSIDE
OUTSIDE are keywords used to specify whether you want the quantity label to be placed inside of the segment or outside of it. If the label is to be placed inside of the segment, make sure any colors specified by attribute_id contrast with the color of the segment.

Location specifies a specific location where you want the percent label to be located, rather than having the location inside or outside chosen for you. The syntax for Location is listed in section 4.3.4 of this manual.

attribute_id is an attribute to be associated with the text of the quantity label. The attribute_id can be either one which is pre-defined or one which the programmer defines using the ATTRIBUTE clause described in section 4.3.1.1.3 of this manual. A list of pre-defined attribute ids is contained in section 4.3.10 of this manual.

PIE Clause, SHADE COLOR Phrase

4.3.2.11.6 SHADE COLOR Phrase

The SHADE COLOR phrase is used to specify a fill color for the segment. SHADE COLOR is optional. If SHADE COLOR is not specified, a color will be chosen based upon the background color and the last color chosen for a fillarea. The syntax for the SHADE COLOR phrase is:

SHADE COLOR color

SHADE COLOR are the phrase keywords.

color is the name of the color with which to fill the segment. (If inside labels are specified, make sure the label color contrasts with the segment color.) Available colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

2-D Graphics Field Definitions

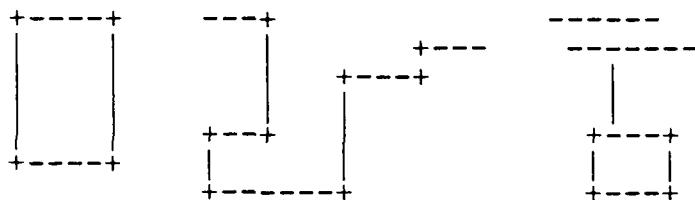
4.3.3 Two Dimensional (2-D) Graphics Field Definitions

The following sections contain the syntax for the definition of two dimensional graphics fields. These fields types include polylines, polymarkers, fillareas, and graphics text. These graphics elements are defined as fields on forms just as are item fields, form fields, window fields, and graph fields.

POLYLINE Field Definition

4.3.3.1 POLYLINE Field Definition

A polyline is a geometric primitive consisting of one or more connected line segments. For example, the figures pictured below are all polylines:



The syntax for defining a polyline is:

```

POLYLINE polyline_id
  [
    {
      STYLE { int SOLID
              DASHED
              DOTTED
              ( DASHED DOTTED ) }
    }
  ]
  [
    SIZE real
  ]
  [
    COLOR color
  ]
  POINTS G_location ...
  [
    APPEARS IF Expression
  ]

```

POLYLINE Statement

4.3.3.1.1 POLYLINE Statement

This statement specifies that the field being defined is to be a polyline. The syntax for the POLYLINE statement is:

POLYLINE polyline_id

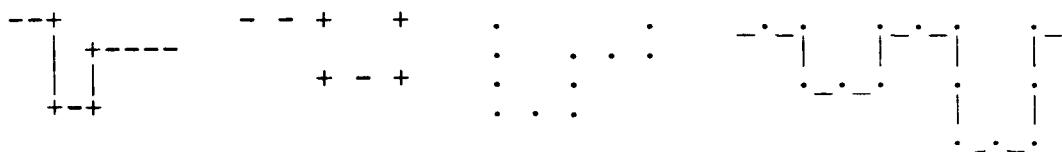
POLYLINE is the clause keyword.

polyline_id is a unique name of up to 10 letters, numbers, and/or underscores. A polyline_id is required and cannot begin with a number.

POLYLINE Statement, STYLE Clause

4.3.3.1.2 STYLE Clause

The STYLE clause is used to specify the type of line you want. You may specify SOLID, DASHED, DOTTED, or DASHED DOTTED. In addition, you may specify an integer for line type. You may specify any integer in the range 1 to 4, inclusive. The integers are simply a numeric way of specifying the same 4 line types above. Below is a pictorial representation of each line type, the name of the type, and the corresponding integer:



SOLID (1) DASHED (2) DOTTED (3) DASHED DOTTED (4)

STYLE is not a required clause. If the STYLE clause is omitted, the polyline will be SOLID. The syntax for the STYLE clause is:

```
STYLE { int
        { SOLID
          { DASHED
            { DOTTED
              { DASHED DOTTED } } } } }
```

STYLE is the clause keyword.

int is an integer from 1 to 4, inclusive, which may be substituted for one of the 4 available line types.

SOLID is a keyword which specifies that the polyline is to be a solid line. SOLID may be represented by the integer 1.

DASHED is a keyword which specifies that the polyline is to be a dashed line. DASHED may be represented by the integer 2.

POLYLINE Statement, STYLE Clause

DOTTED is a keyword which specifies that the polyline is to be a dotted line. DOTTED may be represented by the integer 3.

DASHED DOTTED are keywords which specify that the polyline is to be a series of alternating dashes and dots. DASHED DOTTED may be represented by the integer 4.

POLYLINE Statement, SIZE Clause

4.3.3.1.3 SIZE Clause

The SIZE clause is used to define the thickness of the polyline. The syntax for the SIZE clause is:

SIZE real

SIZE is the clause keyword.

real is a real number which represents the thickness of the line. 1 represents a normal line thickness, 2 represents a line twice the normal thickness, .5 represents a line half the normal thickness, and so on. ("Normal" means whatever the default is for the terminal on which you are operating.)

POLYLINE Statement, COLOR Clause

4.3.3.1.4 COLOR Clause

The COLOR clause is used to specify the color of the polyline. The COLOR clause is not required. If it is omitted, the default will be the last color which was assigned to a polyline. If no color has yet been assigned to a polyline on the form, the default color will be chosen based upon the background color of the form. The syntax for the COLOR clause is:

COLOR color

COLOR is the clause keyword.

color is the name of the color you wish to assign to the polyline. Valid colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

POLYLINE Statement, POINTS Clause

4.3.3.1.5 POINTS Clause

The POINTS clause is used to specify the points that will be used to make up the polyline. In the following diagram, integers will be used to mark the minimum number of points and the order in which they must be specified to make up the polylines shown:

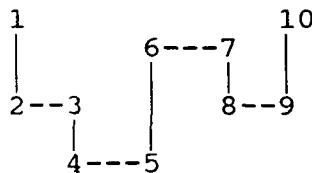


Figure A

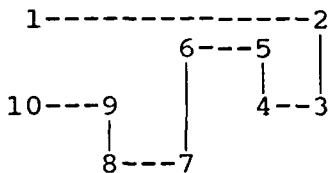


Figure B

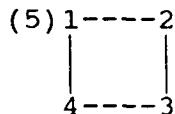


Figure C

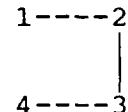


Figure D

The line is drawn starting at the first point defined, and proceeding to each subsequent point in the order specified. Notice that Figure A and Figure B have the exact same points, yet they are specified in a different order, so the figures are different. Figures C and D are the same except that the first point was specified again as the last point in Figure C, resulting in a closed figure. If the first point is not re-specified, the result will be an open figure like Figure D.

The syntax for the POINTS clause is:

POINTS G_location ...

POINTS is the clause keyword.

G_location is the location of one of the points. You may specify multiple locations in order to specify multiple points. The line will be drawn from the first point specified to the second point, and then to the third, and so on, in order, until all points specified have been connected by the line. The syntax for G_location is described in section 4.3.5 of this manual.

POLYLINE Statement, APPEARS IF Clause

4.3.3.1.6 APPEARS IF Clause

The APPEARS IF clause is used to specify when the polyline is to appear on its containing form. If the Expression parameter evaluates to true, the polyline will appear. If it evaluates to false, the polyline will not appear. If the APPEARS IF clause is omitted, the polyline will always appear on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

APPEARS IF are the clause keywords.

Expression is the truth-valued criterion that determines when the polyline appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

POLYMARKER Field Definition

4.3.3.2 POLYMARKER Field Definition

A polymarker is a geometric primitive consisting of one or more marker symbols (such as a cross or a dot). For example, the following are polymarkers:

o + . * x

The syntax for defining a polymarker is:

```
POLYMARKER polymarker_id
  {
    STYLE { int } }
    DOT
    PLUS
    STAR
    CIRCLE
    CROSS
  }
  [
    SIZE real
  ]
  [
    COLOR color
  ]
  POINTS G_location ...
  [
    APPEARS IF Expression
  ]
```

POLYMARKER Statement

4.3.3.2.1 POLYMARKER Statement

This statement specifies that the field being defined is to be a polymarker. The syntax for the POLYMARKER statement is:

POLYMARKER polymarker_id

POLYMARKER is the clause keyword.

polymarker_id is a unique name of up to 10 letters, numbers, and/or underscores. A polymarker_id is required and cannot begin with a number.

POLYMARKER Statement, STYLE Clause

4.3.3.2.2 STYLE Clause

The STYLE clause is used to specify the type of marker you want. You may specify DOT, PLUS, STAR, CIRCLE, or CROSS. In addition, you may specify an integer for marker type. You may specify any integer in the range 1 to 5, inclusive. The integers are simply a numeric way of specifying the same 5 marker types listed above. Below is a pictorial representation of each marker type, the name of the type, and the corresponding integer:

.	+	*	o	x
(1)	(2)	(3)	(4)	(5)
DOT	PLUS	STAR	CIRCLE	CROSS

STYLE is not a required clause. If the STYLE clause is omitted, the polymarker will be DOTs. The syntax for the STYLE clause is:

```
[ { int      ) } ]  
  STYLE { DOT  
          PLUS  
          STAR  
          CIRCLE  
          CROSS  
          ( ) } ]
```

STYLE is the clause keyword.

int is an integer from 1 to 5, inclusive, which may be substituted for one of the 5 available marker types, as illustrated in the diagram above.

DOT specifies that the marker type desired is a dot (.).

PLUS specifies that the marker type desired is a plus sign (+).

STAR specifies that the marker type desired is an asterisk, or star (*).

POLYMARKER Statement, STYLE Clause

CIRCLE specifies that the marker type desired is a circle (o).

CROSS specifies that the marker type desired is a cross (x).

POLYMARKER Statement, SIZE Clause

4.3.3.2.3 SIZE Clause

The SIZE clause is used to define the height of the polymarker (width need not be specified; it will be expanded proportionally to the height automatically). The syntax for the SIZE clause is:

SIZE real

SIZE is the clause keyword.

real is a real number which represents the height of the marker. 1 represents a normal character height, 2 represents a marker twice the normal character height, .5 represents a line half the normal height, and so on. ("Normal" means whatever the default is for the terminal on which you are operating.) Width is not specified. Width will be automatically expanded proportionally to the height selected, so that whatever height/width ratio is supported by the terminal is kept constant.

POLYMARKER Statement, COLOR Clause

4.3.3.2.4 COLOR Clause

The COLOR clause is used to specify the color of the polymarker. The COLOR clause is not required. If it is omitted, the default will be the last color which was assigned to a polymarker. If no color has yet been assigned to a polymarker on the form, the default color will be chosen based upon the background color of the form. The syntax for the COLOR clause is:

COLOR color

COLOR is the clause keyword.

color is the name of the color you wish to assign to the polymarker. Valid colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

POLYMARKER Statement, POINTS Clause

4.3.3.2.5 POINTS Clause

The POINTS clause is used to specify the location(s) at which you wish a polymarker to appear. The syntax for the POINTS clause is:

POINTS G_location ...

POINTS is the clause keyword.

G_location is the location of one of the points at which you wish to have a polymarker appear. You may specify multiple locations in order to specify multiple points. The syntax for G_location is described in section 4.3.5 of this manual.

POLYMARKER Statement, APPEARS IF Clause

4.3.3.2.6 APPEARS IF Clause

The APPEARS IF clause is used to specify when the polymarker is to appear on its containing form. If the Expression parameter evaluates to true, the polymarker will appear. If it evaluates to false, the polymarker will not appear. If the APPEARS IF clause is omitted, the polymarker will always appear on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

APPEARS IF are the clause keywords.

Expression is the truth-valued criterion that determines when the polymarker appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

Graphics Text Field Definition

4.3.3.3 Graphics Text Field Definition

A graphics text field is a graphics field which contains graphical representations of textual characters. Graphics text is a geometric primitive consisting of a number of characters with a particular orientation arranged along a particular path and aligned in a particular manner with some point. Graphics text can be expanded or compressed in size, the colors can be changed, it can run in different directions (left to right, right to left, vertically, diagonally, etc.), it can begin in the middle of a character cell. The syntax for producing a graphics text field is:

```
GTEXT gtext_id
  [ EXPANSION real ]
  [ SPACING real ]
  [ SIZE real ]
  [ COLOR color ]
  {
    PATH { RIGHT
          LEFT
          UP
          DOWN
        }
  }
G_location string
  [ APPEARS IF Expression ]
```

GTEXT Statement

4.3.3.3.1 GTEXT Statement

The GTEXT is used to create a field which will contain a graphical representation of textual information. Graphics text can be changed in size, orientation (horizontal, vertical), and color. By manipulating the size of the characters using the SPACING, EXPANSION, and SIZE clauses, it is possible to display more than 80 characters on a line on a terminal which is only 80 characters wide. The syntax for the GTEXT statement is:

GTEXT text_id

GTEXT is the clause keyword.

text_id is a unique name of up to 10 letters, numbers, and/or underscores. A **text_id** is required and cannot begin with a number.

GTEXT Statement, EXPANSION Clause

4.3.3.3.2 EXPANSION Clause

The EXPANSION Clause is used to change the width of the graphics text character. EXPANSION does not effect the height of the character. The syntax for the EXPANSION clause is:

EXPANSION real

EXPANSION is the clause keyword.

real is a real number which represents the width of the graphics character. 1 represents the normal width of a character, 2 represents a character twice the normal width, .5 represents a character one half the normal width, and so on. ("Normal" means whatever the default is for the terminal on which you are operating.)

GTEXT Statement, SPACING Clause

4.3.3.3.3 SPACING Clause

The SPACING Clause allows you to change the size of the gap between graphics characters (i.e., the spaces between the letters). The syntax for the SPACING clause is:

SPACING real

SPACING is the clause keyword.

real is a real number which represents the size of the gap between the characters. 1 represents a normal sized gap, 2 represents a gap twice the normal size, .5 represents a gap one half the normal size, and so on. ("Normal" means whatever the default is for the terminal on which you are operating.)

GTEXT Statement, SIZE Clause

4.3.3.3.4 SIZE Clause

The SIZE clause is used to specify the height of the graphics text characters. The width of the character will remain proportional to the height. The syntax for the SIZE clause is:

SIZE real

SIZE is the clause keyword.

real specifies the height of the graphics character. 1 represents a character of normal height, 2 represents a character which is twice the normal height, .5 represents a character which is one half the normal height, and so on. ("Normal" means whatever the default is for the terminal on which you are operating.)

GTEXT Statement, COLOR Clause

4.3.3.3.5 COLOR Clause

The COLOR clause is used to specify the color of the text field. The COLOR clause is not required. If it is omitted, the default will be the last color which was assigned to a text field. If no color has yet been assigned to a text field on the form, the default color will be chosen based upon the background color of the form. The syntax for the COLOR clause is:

COLOR color

COLOR is the clause keyword.

color is the name of the color you wish to assign to the text field. Valid colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

GTEXT Statement, PATH Clause

4.3.3.3.6 PATH Clause

The PATH clause is used to specify the orientation, or direction, of the graphics text you are defining. Graphics text can be defined to run left to right, right to left, top to bottom, or bottom to top, as illustrated in the following diagram:

TEXT	TXET	T	T
		E	X
		X	E
		T	T

(right) (left) (down) (up)

The syntax for the PATH clause is:

PATH {
 RIGHT
 LEFT
 UP
 DOWN
}

PATH is the clause keyword.

RIGHT is a keyword which specifies that the graphics text is to run left to right (e.g., T E X T). If the PATH clause is omitted, the default will be RIGHT.

LEFT is a keyword which specifies that the graphics text is to run right to left (e.g., T X E T).

UP is a keyword which specifies that the graphics text is to run bottom to top (e.g., T
 X
 E
 T).

DOWN is a keyword which specifies that the graphics text is to run top to bottom (e.g., T
 E
 X
 T).

GTEXT Statement, G_location Clause

4.3.3.3.7 G_location Clause

The G_location clause is used to specify the exact location where you want the first character of the text field to appear on the form. The G_location clause differs from the Location clause in only one way. The Location clause must be specified in integers, with the integers standing for the number of a character cell. The G_location clause is specified in real numbers, so that you are able to start a graphics field in the middle of a character cell if it is desired.

The G_location clause is also used to specify the string which you want displayed in the GTEXT field. Neither G_location nor string is optional. BOTH must be specified. The syntax for the G_location clause is:

G_location string

G_location is the exact location where you want the first character of the graphics text string to appear. G_location is specified using the syntax described in section 4.3.5 of this manual.

string is the character string you wish to display in the GTEXT field. The text string must be enclosed in double quotes ("display this string").

GTEXT Statement, APPEARS IF Clause

4.3.3.2.6 APPEARS IF Clause

The APPEARS IF clause is used to specify when the GTEXT field is to appear on its containing form. If the Expression parameter evaluates to true, the GTEXT field will appear. If it evaluates to false, the GTEXT field will not appear. If the APPEARS IF clause is omitted, the GTEXT field will always appear on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

APPEARS IF are the clause keywords.

Expression is the truth-valued criterion that determines when the GTEXT field appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

Fillarea Field Definition

4.3.3.4 Fillarea Field Definition

A fillarea is a geometric primitive consisting of a planar area which is to be filled in with a particular color or pattern. A fillarea is defined in much the same manner as a polyline, except that the first point need not be re-specified in order to close the figure. A fillarea is by definition a closed figure. The syntax for defining a fillarea is:

```
FILLAREA fillarea_id
  [ COLOR color ]
  POINTS G_location ...
  [ APPEARS IF Expression ]
```

FILLAREA Statement

4.3.3.4.1 FILLAREA Statement

The FILLAREA statement is used to define a fillarea field on a form. The syntax for the FILLAREA statement is:

FILLAREA fillarea_id

FILLAREA are the statement keywords.

fillarea_id is a unique name of up to 10 letters, numbers, and/or underscores. A fillarea_id is required and cannot begin with a number.

FILLAREA Statement, COLOR Clause

4.3.3.4.2 COLOR Clause

The COLOR clause is used to specify color of the fillarea. The COLOR clause is not required. When COLOR is specified, the fillarea will be shaded in the color specified. If COLOR is omitted, the color of the fillarea will default to the last color used for a fillarea on that form. If no other fillarea exists on that form, the default color will depend upon the background color of the form. The syntax for the COLOR clause is:

COLOR color

COLOR is the clause keyword.

color is the name of the color you wish to assign to the fillarea. Valid colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

FILLAREA Statement, POINTS Clause

4.3.3.4.3 POINTS Clause

The POINTS clause is used to specify the points that will be used to define the fillarea. In the following diagram, an asterisk, or "star", will be used to mark the minimum number of points which must be specified to make up the fillareas shown:

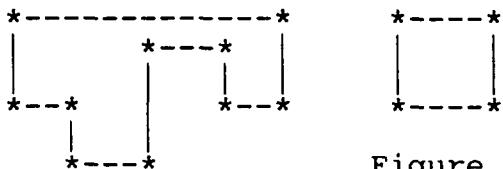


Figure A

The line is drawn starting at the first point defined, and proceeding to each subsequent point in the order specified. Care must be taken to ensure that the points are specified in the correct order. Note that the first point need not be re-specified in order to close the figure. All fillareas are closed figures by definition, so a line is automatically drawn from the last point specified back to the first point.

The syntax for the POINTS clause is:

POINTS G_location ...

POINTS is the clause keyword.

G_location is the location of each of the points. You may specify multiple locations in order to specify multiple points. The line will be drawn from the first point specified to the second point, and then to the third, and so on, in order, until all points specified have been connected by the line. The syntax for G_location is described in section 4.3.5 of this manual.

FILLAREA Statement, APPEARS IF Clause

4.3.3.4.4 APPEARS IF Clause

The APPEARS IF clause is used to specify when the fillarea field is to appear on its containing form. If the Expression parameter evaluates to true, the fillarea field will appear. If it evaluates to false, the fillarea field will not appear. If the APPEARS IF clause is omitted, the fillarea field will always appear on the form. The syntax for the APPEARS IF clause is:

APPEARS IF Expression

APPEARS IF are the clause keywords.

Expression is the truth-valued criterion that determines when the fillarea field appears on its containing form. Valid expression types are explained in section 4.3.7 of this manual.

Location Clause/Parameter

4.3.4 Location Clause/Parameter

Location specifies where text and fields will be positioned on a form. Location may be absolute with respect to the origin of the form or relative to fields on the form. The origin of a form is at the top left corner with rows being positive down and columns positive to the right. Relative locations are especially useful for positioning text that is associated with fields. The syntax for location is:

```
( ( [ int ] ( LEFT ) { field_id } ) ) { AND
{ COLUMN int ( ) ) OF [ ( FORM ) ] ) }
( [ int ] ( RIGHT ) { field_id } ) )
{ COLUMN int ( ) ) OF [ ( FORM ) ] ) )
( [ int ] ( BELOW ) { field_id } ) )
{ ROW int ( ) ) [ ( FORM ) ] )
( [ int ] ( ABOVE ) { field_id } ) )
{ ROW int ( ) ) [ ( FORM ) ] ) { AND
[Rpt] AT { ( [ int ] ( RIGHT ) { field_id } ) )
{ COLUMN int ( ) ) OF [ ( FORM ) ] ) )
( [ int ] ( LEFT ) { field_id } ) )
{ [ int ] ( RIGHT ) OF [ Rpt OF ] [ ( FORM ) ] )
( [ int ] ( ABOVE ) { field_id } ) )
{ [ int ] ( BELOW ) [ Rpt OF ] [ ( FORM ) ] )
( [ int-1 int-2 [ RELATIVE TO [ Rpt OF ] [ field_id ]
[ ( FORM ) ] ] ) )
```

When defining relative locations, field reference points are used. Figure 4-1 shows the nine possible reference points that a field can have.

LOCATION Clause/Parameter

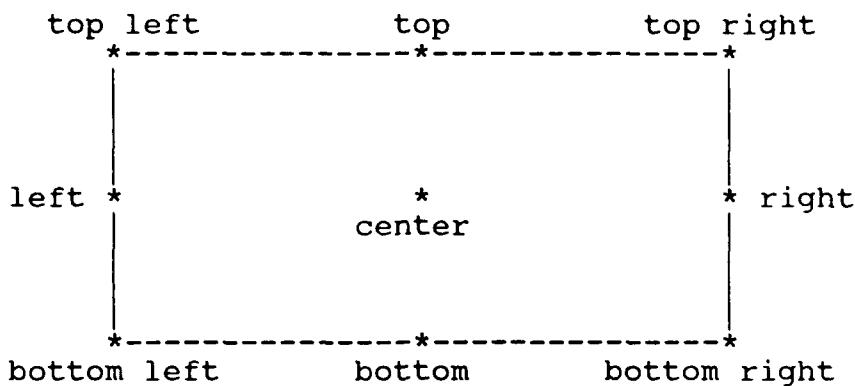


Figure 4-1 Field Reference Points

Each of the reference points represents a character. This means that if you have a one character field, all nine points are the same.

The syntax for the Rpt parameter in the location syntax is:

```
( TOP LEFT
  TOP
  TOP RIGHT
  LEFT
  CENTER
  RIGHT
  BOTTOM LEFT
  BOTTOM
  ( BOTTOM RIGHT )
```

When positioning text and fields, they must be contained within the boundaries of the containing form and cannot overlap other fields or text. Column one of a form must always be blank and there must be one blank space between an item field and any prompt text. For example:

prompt: _____ is legal and

prompt: _____ is illegal.

LOCATION Clause/Parameter

When the relative field name is omitted in the location syntax, the relative field is assumed to be the parent (i.e., for prompts, the field the prompt is associated with, and for fields, the form which contains the field). In addition, the name "(FORM)" can be used to explicitly refer to the containing form.

The following sections describe and show how to use the location syntax to position fields and text. # represents the form origin and * represents the default field reference point.

4.3.4.1 Absolute Location

An absolute location positions the first character of a text string or a reference point of a field at the intersection of row (n) and column (m) with respect to the form origin. Both coordinates must be given. When positioning a field, the default reference point is the top left if the reference point is not given.

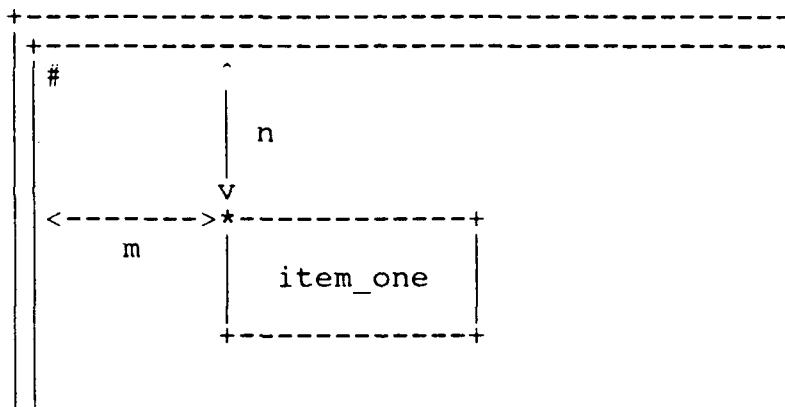


Figure 4-2 Absolute Location

LOCATION Clause/Parameter

The reference point of field_one (*) is positioned at the intersection of row n and column m. To position field_one as shown, you would use the location syntax:

[Rpt] AT COLUMN int AND ROW int

or

[Rpt] AT ROW int AND COLUMN int

or

[Rpt] AT int-1 int-2 [RELATIVE TO [Rpt OF]
[field_id]]

where int-1 is the row and int-2 is the column. Some valid Locations are:

TOP LEFT AT COL m AND ROW n

TOP LEFT AT n m

AT n m

4.3.4.2 Relative Location

The reference point of a field can be positioned at a point that is relative to the reference point of another field on the form. If a specific reference point for either of the fields is not given, the default is the top left.

LOCATION Clause/Parameter

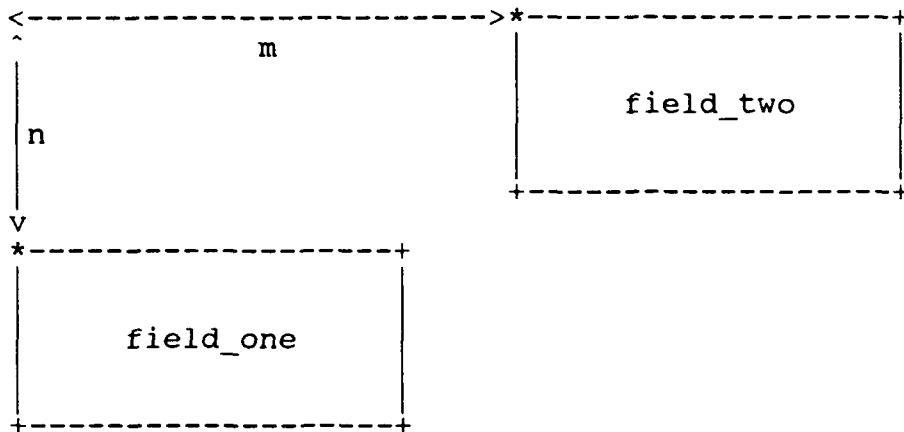


Figure 4-3 Relative Location

The reference point of `field_two` is positioned at row `n` and column `m` relative to the reference point of `field_one`. To position `field_two` as shown, you would use the location syntax:

`[Rpt] AT int-1 int-2`

where `int-1` is the row and `int-2` is the column. Some valid locations are:

`TOP LEFT AT -n m RELATIVE TO TOP LEFT OF field_one`

`AT -n m RELATIVE TO field_one`

4.3.4.2.1 Relative Location (Above/Below)

The reference point of a field can be positioned `n` rows above or below the reference point of another field on the form. If reference points are not given for either of the fields, the default reference points are as shown by the *s.

LOCATION Clause/Parameter

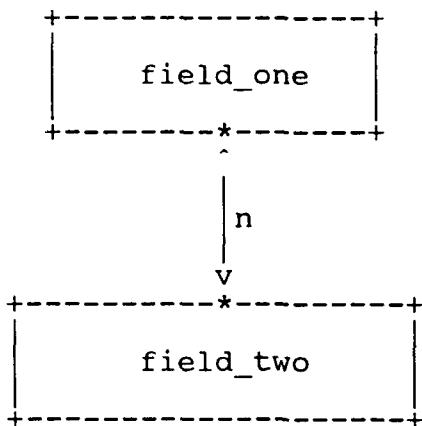


Figure 4-4 Relative Location (Above/Below)

The reference point of `field_two` is positioned `n` rows below the reference point of `field_one`. To position `field_two` as shown, you would use the location syntax:

```
[ Rpt ] AT [ int ] { ABOVE } [ Rpt OF ] [ field_id ]  
          { BELOW } [ ]
```

Some valid Location clauses are:

`TOP AT n BELOW BOTTOM OF field_one`

`AT n BELOW field_one`

Using the `ABOVE` keyword, you can position the reference point of `field_one` `n` rows above the reference point of `field_two`. Some valid Locations are:

`BOTTOM AT n ABOVE TOP OF field_two`

`AT n ABOVE field_two`

4.3.4.2.2 Relative Location (Right/Left)

The reference point of a field can be positioned `m` columns right or left of the reference point of another field on the form. If reference points are not given for either of the fields, the default points are as shown by the `*`s.

LOCATION Clause/Parameter

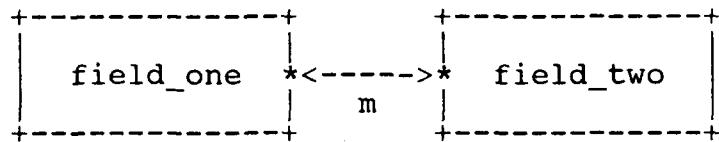


Figure 4-5 Relative Location (Right/Left)

The reference point (Rpt) of field_one is m columns to the left of the field_two reference point or the field_two reference point is m columns to the right of the field_one reference point. To position the fields as shown, you would use the location syntax:

```
[ Rpt ] AT [ int ] { LEFT } OF [ Rpt OF ] [ field_id ]  
          ( RIGHT )
```

Some valid Locations are:

RIGHT AT m LEFT OF LEFT OF field_two

AT m LEFT OF LEFT OF field_two

LEFT AT m RIGHT OF RIGHT OF field_one

AT m RIGHT OF RIGHT OF field_one

AT m RIGHT OF field_one

NOTE: The Rpt is optional so locations 1 and 2 shown above are identical as are locations 3 and 4.

4.3.4.2.3 Location Relative to Two Fields

The reference point of a field can be positioned n rows above or below the default reference point of one field and m columns right or left of the default reference point of another field. Any reference point can be specified for the field being positioned. The reference points for the other fields default to the edge of the field closest to the field being positioned as shown by the *s.

LOCATION Clause/Parameter

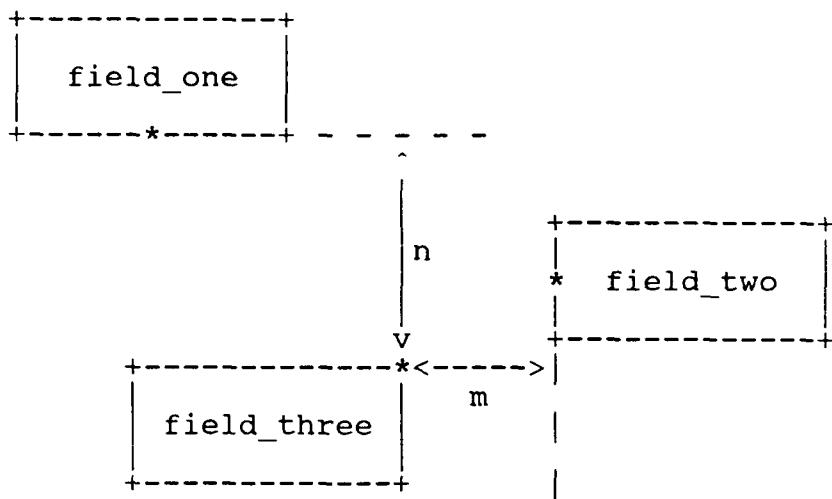


Figure 4-6 Location Relative to Two Fields

The reference point of field_three is positioned n rows below the bottom edge of field_one and m columns left of the left edge of field_two. To position field_three as shown, you would use the location syntax:

```
[ Rpt ] AT [ int ] { ABOVE } [ field_id ]
          ( )
          ( LEFT )
AND [ int ] { RIGHT } OF [ field_id ]
          ( )
```

It does not matter which direction is specified first. Some valid Locations are:

```
TOP RIGHT AT n BELOW field_one AND m LEFT OF field_two
AT m LEFT field_two AND n BELOW field_one
AT n BELOW field_one AND m LEFT field_two
```

LOCATION Clause/Parameter

4.3.4.3 Combination Location

Field positions can be a combination of absolute and relative locations. The reference point of a field can be positioned at row n and m columns right or left of the default reference point of another field or at column m and n rows above or below the default reference point of another field. Any reference point can be specified for the field being positioned. The reference point for the other field defaults to the edge of the field closest to the field or text being positioned as shown by the *s.

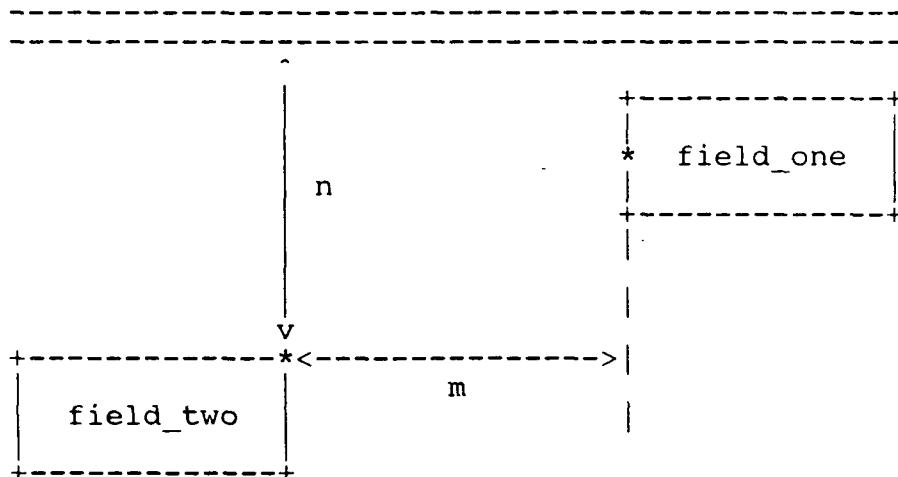


Figure 4-7 Absolute Row/Relative Column

The field_two reference point is in row n, m columns left of the left edge of field_one. To position field_two as shown, you would use the location syntax:

[Rpt] AT ROW int AND [int] (LEFT)
[Rpt] AT ROW int AND [int] (RIGHT) OF [field_id]
()

or

[Rpt] AT [int] (LEFT)
[Rpt] AT [int] (RIGHT) OF [field_id] AND ROW int
()

LOCATION Clause/Parameter

Some valid Locations are:

TOP RIGHT AT ROW n AND m LEFT OF field_one

AT m LEFT field_one AND ROW n

AT ROW n AND m LEFT field_one

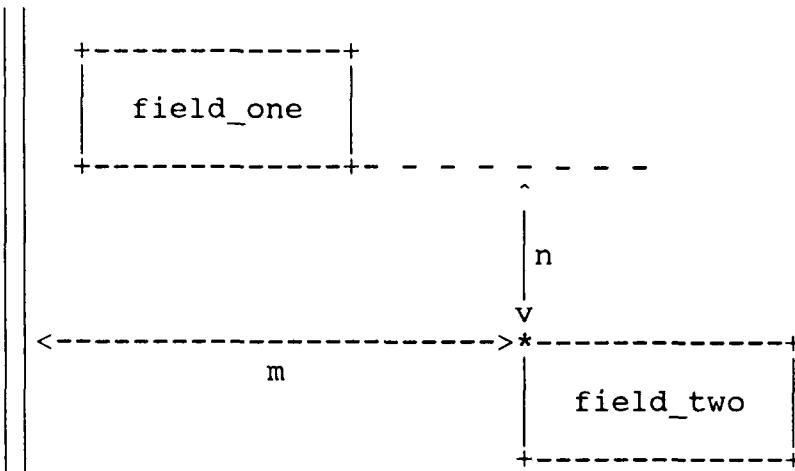


Figure 4-8 Absolute Column/Relative Row

The field_two reference point is in column m, n rows below the bottom edge of field_one. To position field_two as shown, you would use the location syntax:

```
[ Rpt ] AT [ int ] { ABOVE } [ field_id ] AND COLUMN int
[ Rpt ] AT [ int ] { BELOW } [ field_id ]
```

or

```
[ Rpt ] AT COLUMN int AND [ int ] { ABOVE } [ field_id ]
[ Rpt ] AT COLUMN int AND [ int ] { BELOW } [ field_id ]
```

LOCATION Clause/Parameter

Some valid locations are:

TOP LEFT AT n BELOW field_one AND COL m

AT TOP LEFT n BELOW field_one AND COL m

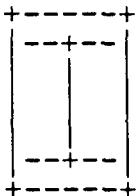
AT COL m AND n BELOW field_one

G_location Clause/Parameter

4.3.5 G location Clause/Parameter

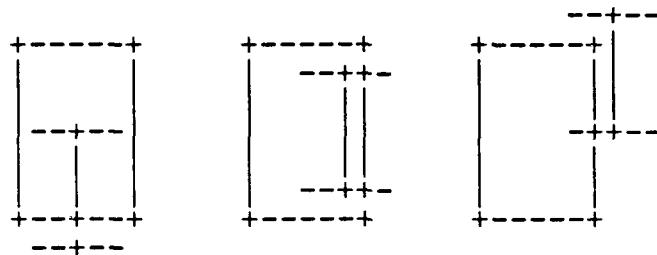
G_location specifies where graphics fields (polylines, polymarkers, graphics text, and fillareas) will be positioned on a form. G_location follows the same rules as location for non-graphics fields with the exception that G_location is specified in real numbers rather than integers. In this way, graphics fields may be located starting in some fraction of a character cell rather than centered in the character cell.

A video terminal screen is made up of several character cells. Most (though not all) terminal screens are 80 cells wide by 24 cells deep. These character cells are, in turn, made up of even smaller divisions called pixels. The number of pixels per character cell also varies by terminal type. Under most circumstances, only one character may fit in one cell at a time. IISS graphics allow the programmer to change the size of a character and to change the placement of a character within a character cell. We can do this because we place graphics characters by pixel instead of by character cell. For example, consider the following figure:



Pictured is a character cell which contains the capital letter I. This is the normal placement of a non-graphics character in the character cell. A graphics character I can be placed in a character cell the same way if the location is specified with integers. If, however, you specify the location using real numbers instead, you may use the fractions to change the placement of the character in the cell. Some possibilities are illustrated below:

G_location Clause/Parameter



The syntax for G_location is:

```
[Rpt] AT { ( ( [ real ] ( LEFT ) { RIGHT } OF [ field_id ] ) ) [ AND
{ COLUMN real ( ) ) ] ) [ ( [ real ] ( BELOW ) { field_id ] ) )
{ ABOVE } [ ( FORM ) ] ) ]
( ROW real ( ) [ ( FORM ) ] ) ]
( [ real ] ( ABOVE ) { field_id ] ) ) [ AND
{ BELOW } [ ( FORM ) ] ) ]
( ROW real ( ) [ ( FORM ) ] ) ]
( [ real ] ( RIGHT ) { LEFT } OF [ field_id ] ) )
( [ real ] { RIGHT } OF [ Rpt OF ] [ field_id ] )
( [ real ] { ABOVE } [ Rpt OF ] [ field_id ] )
( [ real ] { BELOW } [ Rpt OF ] [ field_id ] )
( real-1 real-2 [RELATIVE TO [Rpt OF] [ field_id ] ] ) }
```

G_location functions in the same way as non-graphics locations, including the use of relative positions. You may refer to section 4.3.5 for further explanation of locations.

Repeat Spec Parameter

4.3.6 Repeat Spec Parameter

The repeat specification specifies that the field appears on the form more than once and whether or not the area can be scrolled. A field may repeat, either horizontally or vertically, n times with m spaces between repetitions to form rows or columns. The repeat specification may then be repeated to form two dimensional arrays of fields. When an array needs to be scrolled, the number of actual data occurrences and how many occurrences should be displayed at one time are both specified. In addition to fixed size arrays, you can also define open-ended arrays. This means that the amount of available data determines the array size. This is a very valuable feature for program optimization. The syntax for the Repeat Specification is:

```
( int-1
  { int-1/int-2 } ( HORIZONTAL )
  { int-1/int-1 } ( VERTICAL ) [ WITH int-3 SPACES ] [ , ... ]
  { int-1/ *
  { * } )
```

int-1 specifies how many times to repeat the field on the form (i.e., display size).

int-1/int-2 indicates that the field is scrollable using the function keys in the "scroll/page" mode of the keyboard. Int-1 is how many times to repeat the field on the form and int-2 is the actual number of times the field occurs.

int-1/int-1 indicates that the field is scrollable using the function keys in the "scroll/page" mode of the keyboard but the keys are passed back to the application program. Int-1 is how many times to repeat the field on the form.

int-1/ * indicates a scrollable open-ended array.

* specifies that the field repeats indefinitely on the form.

HORIZONTAL states that the field repeats in a horizontal direction. The abbreviation H may be used for this option.

Repeat Spec Parameter

VERTICAL	states that the field repeats in a vertical direction. The abbreviation V may be used for this option.
WITH	is an optional reserved word that may be included for readability.
int-3	is how many spaces to leave between field occurrences. If int-3 is omitted, the default is one.
SPACES	is an optional reserved word that may be included for readability.

4.3.6.1 One Dimensional Array

A one dimensional array can be defined to repeat either horizontally or vertically. The Repeat Spec (3 H 1) specifies a field which repeats three times horizontally with one space between repetitions. It will appear as shown in Figure 4-9 when displayed to the user:

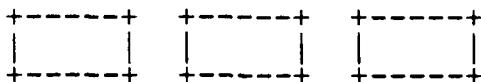


Figure 4-9 One Dimensional Array

4.3.6.2 Two Dimensional Array

A two dimensional array can be defined using the Repeat Spec parameter. The Repeat Spec (2 V 1, 3 H 1) specifies a field which repeats the one dimensional array defined by '3 H 1' two times vertically with one blank row between repetitions. It will appear as shown in Figure 4-10 when displayed to the user.

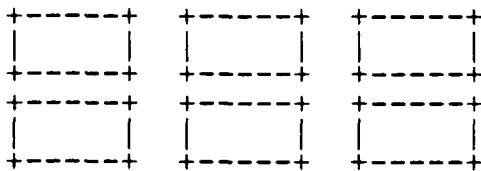


Figure 4-10 Two Dimensional Array

Repeat Spec Parameter

4.3.6.3 Three Dimensional Array

Dimensions can be added to arrays indefinitely. The Repeat Spec (2 V 5, 2 V 1, 3 H 1) specifies repeating the array of fields defined by '2 V 1, 3 H 1' two times vertically with five blank rows between repetitions. It will appear as shown in Figure 4-11 when displayed to the user.

Figure 4-11 Three Dimensional Array

4.3.6.4 One Dimensional Scrolled Array

The Repeat Spec (3/5 HORIZONTAL WITH 2 SPACES) or (3/5 H 2) creates an array with five occurrences, three of which are displayed on the screen horizontally with two spaces between each occurrence. It will appear as shown in Figure 4-12 when displayed to the user:

Repeat Spec Parameter

1	2	3	4	5

Figure 4-12 One Dimensional Scrolled Array

The fields which are actually displayed are enclosed in boxes. The scrolling and paging function keys used to display the other fields are explained in the Terminal Operator's Guide.

4.3.6.5 Two Dimensional Scrolled Array

The Repeat Spec (1/2 V 1, 3/5 H 2) repeats the array defined above once vertically with 1 row between occurrences. The actual size of the array is five columns wide and 2 rows deep. It will appear as shown in Figure 4-13 when displayed to the user:

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5

Figure 4-13 Two Dimensional Scrolled Array

The fields which are actually displayed are enclosed in boxes. The scrolling and paging function keys used to display the other fields are explained in the Terminal Operator's Guide.

Expression Parameter

4.3.7 Expression Parameter

The Expression parameter is used with the VALUE clause to specify the default value of a field, the APPEARS IF clause to specify when a field appears on its containing form, or as a condition definition to determine when to take a pre-defined action. Certain of the different possibilities for expression parameter are more appropriate for the VALUE clause than for the APPEARS IF clause or the condition definition, just as some are more appropriate for APPEARS IF and some for the condition definition, although syntactically, it is legal to use any of them wherever it is indicated that an Expression is needed. For example, it would make more sense to say VALUE "example" than it would to say VALUE OVERFLOW('form.item'), just as it is more logical to say APPEARS IF 'form.item' = "example" than to say APPEARS IF 23 .

All expression possibilities are included here. The syntax for expression is:

Expression Parameter

```
{ string
  item_name
  int
  real
  flag_id
  ( Expression )
  - Expression
  Expression || Expression
  Expression + Expression
  Expression - Expression
  Expression * Expression
  Expression / Expression
  Expression < Expression
  Expression <= Expression
  Expression = Expression
  Expression != Expression
  Expression > Expression
  Expression >= Expression
  Expression AND Expression
  Expression OR Expression
  NOT Expression
  Expression ? Expression : Expression
  INDEX(field_name)
  BETWEEN(Expression, Expression,
    Expression)
  IN(Expression, Expression,...)
  GETATT(field_name, type)
  GPAGE(window_name)
  GWINDO(window_name, page)
  APPEARS(field_name)
  CURSOR(field_name)
  ROLE(Expression)
  PICK(Expression)
  MODIFY(field_name)
  STARTUP()
  OVERFLOW(field_name)
  CHANGE(item_name)
  ( EMPTY(item_name)
  )
```

Expression Parameter

String is a character string enclosed in double quotes ("default value"). If the SIZE clause is included in a field definition which has a VALUE clause attached which uses String, the length of the string can be no more than the total number of characters specified by the size. For example, if size is 4 by 3, then the string should be no more than 12 characters long. When entering default values for multi-dimensional fields, concatenate the values and they will be split apart appropriately to fill the field. For example, if the size is 4 by 3 and the string is "*****+====", it will be displayed as:

+++
====

NOTE: Trailing blanks are not considered significant.

item_name is a qualified name enclosed in single quotes ('field') identifying an item field whose value you want to use as a default value or an appears if criterion. (Refer to section 4.5 for a further explanation of qualified names.) The field names '._time' and '._date' may be used to access the current time and date in the formats HH:MM:SS and MM/DD/YY respectively. In this case, the SIZE of the field being defined must be 8 and its display attribute must be calculated. For example:

```
ITEM Time
SIZE 8
DISPLAY AS calculated
VALUE '.time'
```

```
ITEM item_one
SIZE 10
DISPLAY AS output
VALUE 'form1.item2'
```

```
ITEM item_one
SIZE 5
APPEARS IF 'form1.item2' = "YES"
```

Expression Parameter

int is an integer value.

real is a real number value.

flag_id is a user-defined name which contains a boolean valued state indicator which may be used in condition and set expressions.

(Expression) specifies the use of parenthesis for changing the precedence of arithmetic operators. You might want to do this to define an APPEARS IF criterion or a condition definition. For example:

```
9 * ( 'I1' + 'I2' )
(( 'I1' >= 1) ? (( 'I1' = 5) ? 1 : 0) : 1)
```

- Expression specifies the negative of an expression as a value. For example: -n

NOT Expression specifies the logical negation of a boolean expression. You might want to use this to define an APPEARS IF criterion or a condition definition. For example:

```
NOT (item1 = "stop")
```

Expression Binop Expression

specifies default values, appears if criterion, and condition definitions as the result of combining expressions using binary operators. In many cases, these combinations are truth-valued expressions with values of 1 or 0. The binary operators are:

Expression Parameter

```
( || )
| +
| -
| *
| /
| =
| !=
| <
| <=
| >
| >=
| AND
| OR
```

These operators have the usual meanings and precedences. The operator `||` is used for string concatenation and has the same precedence as `+` and `-`. Operands of the wrong type (i.e., character vs. integer) are automatically converted. For example:

```
'I1' > 100
'I1' != "YES"
'I1' <= 0 OR 'I1' >= 10
```

would be valid APPEARS IF clause Expressions or condition definitions, and

```
"User" || " Manual"
```

might be used as a VALUE clause Expression.

Expression ? Expression : Expression

is a conditional assignment expression. If the expression to the left of the `?` evaluates to true then the expression to the right of the `?` is evaluated. If it is false, then the expression to the right of the `:` is evaluated. This expression type is useful in defining APPEARS IF criterion and condition definitions. For example:

Expression Parameter

ITEM item_one
APPEARS IF 'I1' > 10 ? 1 : 0

specifies that item_one appears only if the value of I1 is greater than 10 and

ITEM item_one
APPEARS IF 'I1' != "CCC" ? 1 : 0

specifies that item_one appears only if the value of I1 is not CCC.

INDEX (field_name)

specifies that the value is the name of the first displayed element of the array "field_name". The array must be on the same form as field being defined and must be enclosed in single quotes (i.e., INDEX('myfield')). For example, if "myfield" is a two dimensional array that has been scrolled once horizontally and twice vertically, INDEX('myfield') might return "myfield(2,3)".

GETATT (field_name, type)

specifies the value as an item field's attribute. This expression type is useful in defining APPEARS IF criterion. For example:

APPEARS IF GETATT ('I1', 0) != "INPUT"

specifies that item_one appears only if the current attribute of I1 is INPUT.

GPAGE (field_name, page)

specifies the value as the qualified name of the form in a specified page of a window. This expression type is useful in defining APPEARS IF criterion condition definitions. For example:

ITEM item_one
APPEARS IF GPAGE ('.win3',1) = "ff3"

specifies that item_one appears only if the form ff3 is in page 1 of the window win3.

Expression Parameter

GWINDO (field_name)

specifies the value as the number of pages in a window. This expression type is useful in defining APPEARS IF criterion or condition definitions. For example:

```
ITEM item_one
APPEARS IF GWINDO ('.win3') > 1
```

specifies that item_one appears only if the window win3 contains more than one page.

APPEARS (field_name)

is a boolean expression that is true if the specified field is displayed and false if the field is not displayed. This expression type is useful in defining APPEARS IF criterion or condition definitions. For example:

```
ITEM item_one
APPEARS IF APPEARS ('form1.item2')
```

specifies that item_one appears only if item2 on form1 appears.

BETWEEN (Expression, Expression, Expression)

is a boolean expression that is true if the first expression value is between the value of the second and third expression values and false if it is not. This expression type is useful in defining APPEARS IF criterion and condition definitions. For example:

```
ITEM item_one
APPEARS IF BETWEEN('I1', 1, 10)
```

specifies that item_one appears only if the value of I1 is between 1 and 10.

Expression Parameter

CURSOR (field_name)

is a boolean expression that is true if the cursor is in the specified field and false if it is not. This expression type is useful in defining APPEARS IF criterion and condition definitions. For example:

```
ITEM item_one  
APPEARS IF CURSOR(item_two)
```

specifies that item_one appears only if the cursor is in the field item_two.

IN (Expression, Expression, ...)

is a boolean expression that is true if the first expression value is in the remaining set of expression values. This expression type is useful in defining APPEARS IF criterion and condition definitions. For example:

```
ITEM item_one  
APPEARS IF IN('I1',1,3,5,7,9)
```

specifies that item_one appears only if the value of I1 is an odd number between 1 and 9.

ROLE (Expression)

is a boolean expression that is true if the role of the current logged on user is the specified value and false if it is not. This expression type is useful in defining APPEARS IF criterion. For example:

```
ITEM item_one  
APPEARS IF ROLE("manager")
```

specifies that item_one appears only if the role of the current logged on user is manager.

Expression Parameter

OVERFLOW (field_name)

allows you to specify one or more actions to be performed when the size of a field is exceeded. A repeating field can cause the overflow of its containing field by repetition in either the horizontal or vertical direction. You specify the name of the field that causes the condition action to be triggered. OVERFLOW specifies that one or more actions will occur if the size of a named field is exceeded. Field_name is the qualified name of the field to be evaluated.

CHANGE (item_name)

allows you to define one or more actions to be performed when the value of a named item field changes. If the item field being tested for a changing value contains database values, then it may be appropriate for these values to have been previously sorted so that all similar values are grouped together. This sorting can be achieved by using the "ORDER BY" option of the SELECT statement (see section 2.4.1.6.4 in the Application Generator User's Manual). If the test for a changing value is positive, the actions are taken after the item field is instantiated with the changed value. CHANGE specifies that one or more actions will be performed when the value of a named item field changes.

STARTUP ()

allows you to define one or more actions to be performed at the beginning of the application. This condition is required and should include as one of its actions the presentation of an initial form.

Expression Parameter

PICK (Expression)

allows you to define one or more actions to be performed when the user presses a programmable function key which you defined in the form definition KEYPAD clause.

MODIFY (field_name)

allows you to define one or more actions to be performed when the value of an item field or item fields of a form are modified by the user. The modify condition checks to see if the field has been modified since the last function key pick. The MODIFY condition is different from the CHANGE condition. The CHANGE condition checks for changes in an item field which has been targeted by a select. The MODIFY condition checks for user changes to an item or item fields of a form between function key picks.

EMPTY (item_name)

is true if the select statement to the specified item name returns zero rows from the database.

Non-Graphics Primitives

4.3.8 Non-Graphics Primitives

A primitive is the lowest level of definition for an attribute. Primitives define the characteristics of an attribute. An attribute may be composed of one or more primitives. For example, the pre-defined attribute TEXT is defined with the primitive "guarded" while the attribute ERROR is defined with two primitives, "ored" and "fastblink". Currently available non-graphics primitives are:

```
( BACKGROUND color )
  DISPLAY color
  BOLD
  DIM
  UNDERSCORE
  SLOWBLINK
  FASTBLINK
  HIDDEN
  GUARDED
  REVERSE
  ORED
  NOWRITE
  TABFIELD )
```

BACKGROUND color specifies the color of the background. Valid colors are:

BLACK	RED	GREEN	YELLOW
BLUE	CYAN	WHITE	MAGENTA

DISPLAY color specifies the color of the printing. Valid colors are as listed for BACKGROUND.

NOTE that a terminal that supports color display is required for colors other than black and white. An application that has color forms can be run on a terminal that only displays black and white. In this case, the forms are displayed either normal or reverse depending on the contrast between the background and character colors.

BOLD specifies that text will be displayed at a higher intensity.

DIM specifies that text will be displayed with less intensity.

Non-Graphics Primitives

UNDERSCORE	specifies that text will be underlined.
SLOWBLINK	specifies that text displayed will be flashing slowly on the screen.
FASTBLINK	specifies that text displayed will be flashing quickly on the screen.
HIDDEN	specifies that text contained in the field will not appear on the screen.
GUARDED	specifies that text contained in the field will appear, but that the user will not be able to change it.
REVERSE	specifies reverse video.
ORED	specifies a temporary option that is to be added to the other characteristics rather than replace them. This primitive is used in combination with others.
NOWRITE	specifies that the application program can write to the field.
TABFIELD	specifies that the tab key may be used to place the cursor in the field.

2-D Graphics Primitives

4.3.9 Graphics Primitives

There are currently three types of primitives available for use when creating attributes to apply to business graphs. They are DISPLAY, STYLE, and SYMBOL. The syntax for specifying these primitives is:

```
DISPLAY color
      ( int
        SOLID
        DASHED
        DOTTED
        ( DASHED DOTTED )

      ( int
        DOT
        PLUS
        STAR
        CIRCLE
        CROSS
        )

SYMBOL { [ SYMBOL FREQUENCY ] frequency }
```

DISPLAY is a keyword currently used for specifying a color to be used as an attribute. An example would be:

```
ATTRIBUTE redatt ( DISPLAY red )
```

Available colors are:

BLACK	RED	GREEN	YELLOW
BLUE	MAGENTA	CYAN	WHITE

STYLE is a keyword used for specifying a specific style of line. An example would be:

```
ATTRIBUTE dotline ( STYLE dotted )
```

2-D Graphics Primitives

Available line styles are:

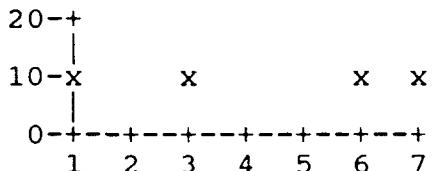
- - - - -	DASHED
.	DOTTED
-----	SOLID
-. -.-.-.-	DASHED DOTTED

An integer may be specified in place of the style name. A solid line is specified with 1, a dashed line with 2, a dotted line with 3, and a dashed dotted line with 4.

SYMBOL Is a keyword used to create an attribute for use with polymarkers. Use this primitive to specify both the type of marker desired and the frequency of appearance for the marker. For example, the following attribute:

```
ATTRIBUTE xmark MARKER ( CROSS FREQ 3 )
```

defines a marker which will be a cross (x). The marker is to appear on every third data point. When this attribute is associated with a graph for the data points (1, 10), (2, 10), (3, 10), (4, 10), (5, 10), (6, 10), (7, 10), the following is produced:



NOTE that the points marked include every third point (3 and 6), AND the first and last points in the set. The first and last points always receive a marker regardless of their value.

NOTE: SYMBOL_FREQUENCY may be abbreviated FREQ.

2-D Graphics Primitives

The available marker styles are:

DOT	.
PLUS	+
STAR	*
CIRCLE	o
CROSS	x

An integer may be specified in place of the marker name in the following manner:

1	DOT
2	PLUS
3	STAR
4	CIRCLE
5	CROSS

Pre-defined Attributes

4.3.10 Pre-defined Attributes

Just as a primitive defines the characteristics of an attribute, the attribute is a way of bundling primitives together under one name and applying them to a field. An attribute can be made up of one or more primitives. The following attribute_ids are pre-defined but may be redefined if desired:

<u>Attribute id</u>	<u>Associated Primitive(s)</u>
ERROR	ored, fastblink
GUARDED	ored, guarded
TEXT	guarded
INPUT	reverse, tabfield
OUTPUT	bold, guarded
HIDDEN	hidden, reverse, tabfield
CALCULATED	bold, guarded, nowrite
TABFLD	guarded, tabfield
XPARNT	transparent (the field assumes the color of the containing form or window)
BLACK	background black, display white
RED	background red, display white
GREEN	background green, display black
YELLOW	background yellow, display black
BLUE	background blue, display white
MAGENTA	background magenta, display white
CYAN	background cyan, display black
WHITE	background white, display black

4.4 Qualified Names

Qualified names are used to refer to Form Processor fields. They are enclosed in single quotes (' '). Qualified names may refer only to those fields contained in the source file. Forms referenced in qualified names must also be presented at some point in the application.

Two symbolic array indices are available. The use of a symbolic index implies a control flow structure loop and at application run-time the symbolic index is replaced with an actual value. The part of the qualified name which prefixes a symbolic index has scope from that condition or action to all nested conditions and actions, inclusive. Scope rules are based on an upper-case comparison of the symbolic index prefix. The syntax for the universal ("for all") quantifier index is an asterisk (*) (e.g., 'myarray(*)'). This means all elements of the array will be used sequentially in the condition or action. The "for each" quantifier index is a zero (e.g., 'myarray(0)'). This symbolic index is normally used in a qualified name which is the target of a SELECT action, or a condition or action which is nested in the SELECT.

4.5 Restrictions

Every form definition must begin with the CREATE FORM statement.

There must be at least one space before and after every keyword in the syntax.

A field_name may only be omitted from the Location syntax in a PROMPT clause.

4.6 Abbreviations

Underscores in the FDL syntax indicate reserved words or portions of reserved words that are optional.

4.7 Including Comments

You can include comments in a form definition by enclosing the comment text in /* and */. Comments are treated a spaces by the FDL compiler. For example:

```
CREATE FORM form1 /* main form */
```

4.8 Reserved Words

This is an alphabetized list of the reserved words in the Form Definition Language:

ABOVE	ABS	ABSOLUTE
ADD	ADDITIVE	ALL
AND	AP	APPEARS
APPLICATION	AS	ASC
ASCENDING	ASSIGN	AT
ATTRIBUTE	AVERAGE	AVG
AXIS	BACKGROUND	BAR
BELOW	BETWEEN	BOTTOM
BOX	BY	CALL
CENTER	CHANGE	CNT
COL	COND	CONDITIONAL
COLOR	COLUMN	COUNT
CURVE	DELETE	DESC
DESCENDING	DIFFERENCE	DISPLAY
DISTINCT	DOMAIN	END
ENTER	EVERY	EXIT
EXPLODE	FILL	FINE
FONT	FORM	FREQ
FROM	GRAPH	GRID
H	HELP	HORIZONTAL
IF	IN	INSERT
INSIDE	INTEGER	INTERSECT
INTO	IS	ITEM
KEY	KEYPAD	LABEL
LEFT	LEGEND	LINE
LINEAR	LINE_TYPE	LINE_WIDTH
LOG	LOWER	MARKER

MAX	MAXIMUM	MIN
MINIMUM	MODIFY	MUST
NILL	NODUP	NOSELECT
NOT	NULL	NUM
NUMERIC	OF	ON
OR	ORDER	OUTSIDE
OVERFLOW	PAGE	PATH
PATTERN	PERCENT	PIC
PICTURE	PIE	PRESENT
PROMPT	QUANTITY	REAL
REDISPLAY	RELATIVE	REPORT
RIGHT	ROW	SCALE
SELECT	SET	SHADE
SIGNAL	SIZE	SPACE
SPACES	START	STARTUP
STYLE	SUM	SUMMARY
SYMBOL	SYMBOL_FREQUENCY	TICK
TO	TOP	UNION
UPPER	UPVECTOR	USING
V	VALUE	VALUES
VERSUS	VERTICAL	WHERE
WINDOW	WITH	XOR

SECTION 5

FORM DRIVEN FORM EDITOR

The Form Driven Form Editor (FDFE) is a software tool for interactively defining and maintaining form definitions used in the IISS environment. The FDFE will not correctly process a form definition which contains graphics fields, graph fields, Appears If clauses, or Application Definition Language statements.

5.1 FDFE Functional Organization

The FDFE functions are organized into three levels of tasks. They are called Work Tasks, Edit Tasks, and Field Tasks. You begin an edit session at the work task level. At this level you can perform maintenance functions on FDL source or FD files or move to the edit task level for a specific FDL source file. At the edit task level, you are editing a specific form definition. Again, you can perform maintenance functions on the form definition or move to the field task level for the specified form definition. At the field task level, you can maintain and edit the individual field definitions that define the specified form.

Figure 5-1 maps the FDFE functional organization to the IISS Form Storage Hierarchy.

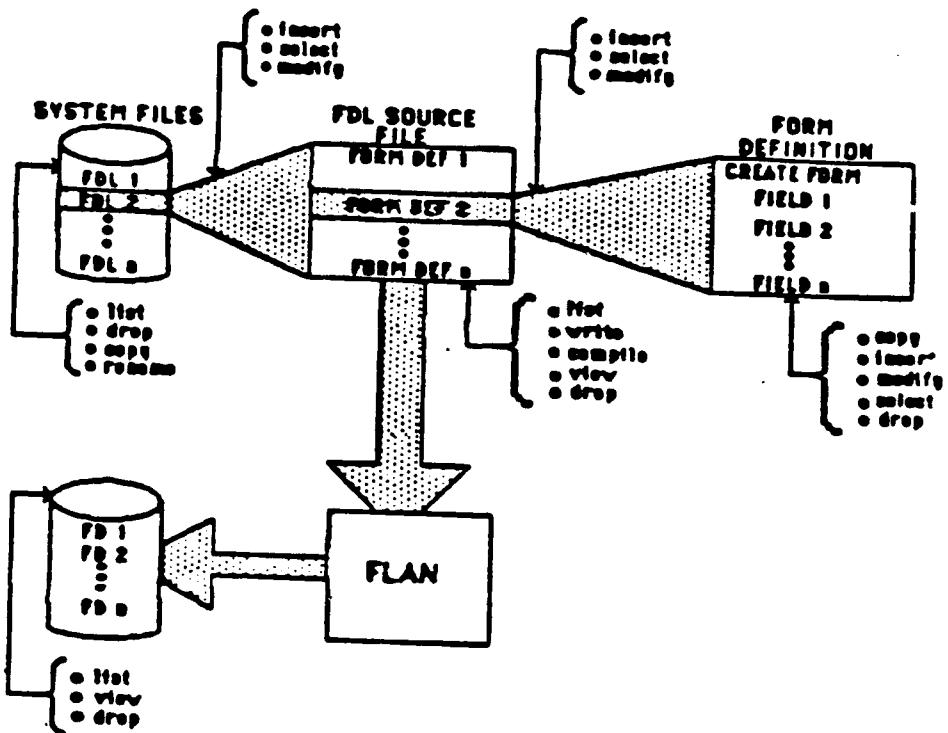


Figure 5-1 FDFE Functions Mapped to the IISS Forms Storage Hierarchy

5.2 FDFE Editing Features

The FDFE provides three modes of operation for editing form definitions. These are Single Field mode, Form mode, and Layout mode. In Single Field mode you edit a form one field at a time. A screen is displayed that provides a template for displaying and defining the characteristics of an individual field. In form mode you edit a form by entering values in a field characteristic table. In layout mode you can graphically position fields and text on the screen and evaluate the overall appearance of a form. You are allowed to switch back and forth between modes to evaluate the results and make changes. When saved, the FDFE stores the form definitions as an FDL source file that can be retrieved and modified.

5.3 System Operation

The FDFE is an application program that can run in the IISS environment. Tasks are performed by entering the appropriate responses to prompts displayed on the screen of the video display terminal. Responses are entered in input fields represented by shaded or underlined areas on the screen. Each time a display appears on the screen, it contains a cursor. The cursor is an underscore or a small flashing rectangle to show where information will appear when entered through the keyboard.

How the user interacts with screens is described in the IISS Terminal Operator Guide. Figure 5-2 shows the layout of the VT100 keypad for the FDFE in the "application" mode of the keyboard.

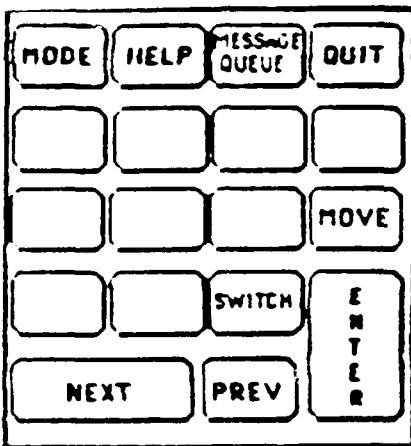


Figure 5-2 VT100 Keypad Layout for the FDFE

5.3.1 Accessing the FDFE

The FDFE is available as an application in the IISS environment as explained in the IISS Terminal Operator Guide. To access the FDFE, enter FDFE as the FUNCTION on the IISS Function Screen. When you have successfully accessed the FDFE, the following display appears on the terminal screen:

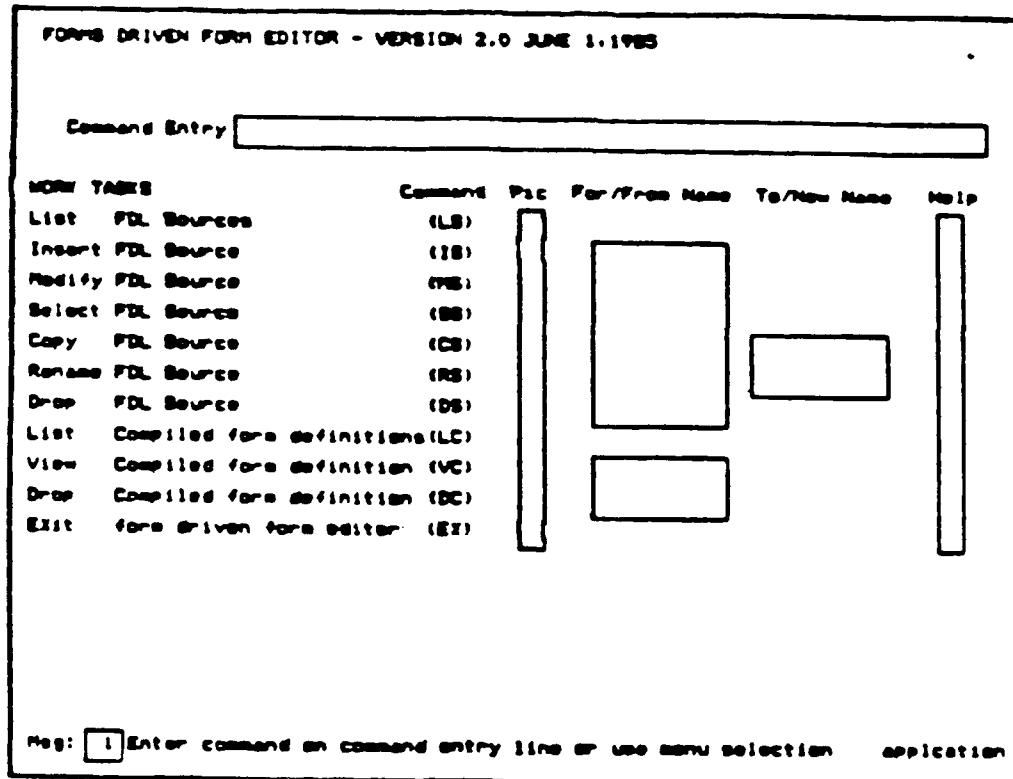


Figure 5-3 Work Task Screen

5.4 Work Task Screen

This screen is a list of the FDFE functions that allow the user to maintain FDL source and FD files and edit a specific FDL source file.

5.4.1 Choosing A Work Task

A work task can be chosen with menu selection or command entry. With menu selection, enter any nonblank character in the "Pic" field for the desired task and the necessary parameters for the task in the input fields that appear in the same horizontal line of the screen. Note that the "Help" field is not a required task parameter. When the <HELP> is pressed key from this field, an explanation of the task is displayed on the screen. An example of choosing a Work Task is shown in the next screen:

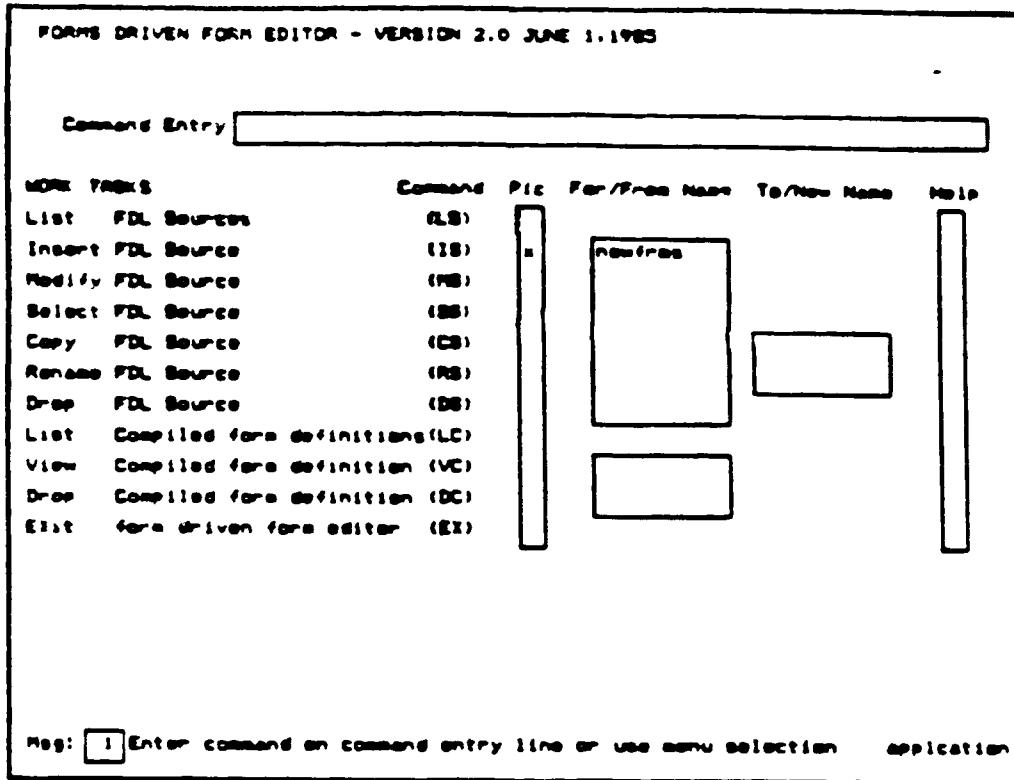


Figure 5-4 Choosing a Work Task-Menu Selection

Note that command entry overrides menu selection and an error message is displayed if more than one menu selection is made.

With command entry, you enter the appropriate two letter task command followed by values for the necessary parameters in the "Command Entry" field and press the <ENTER> key. The parameters must be entered in the order specified by menu selection and separated by blanks. For example:

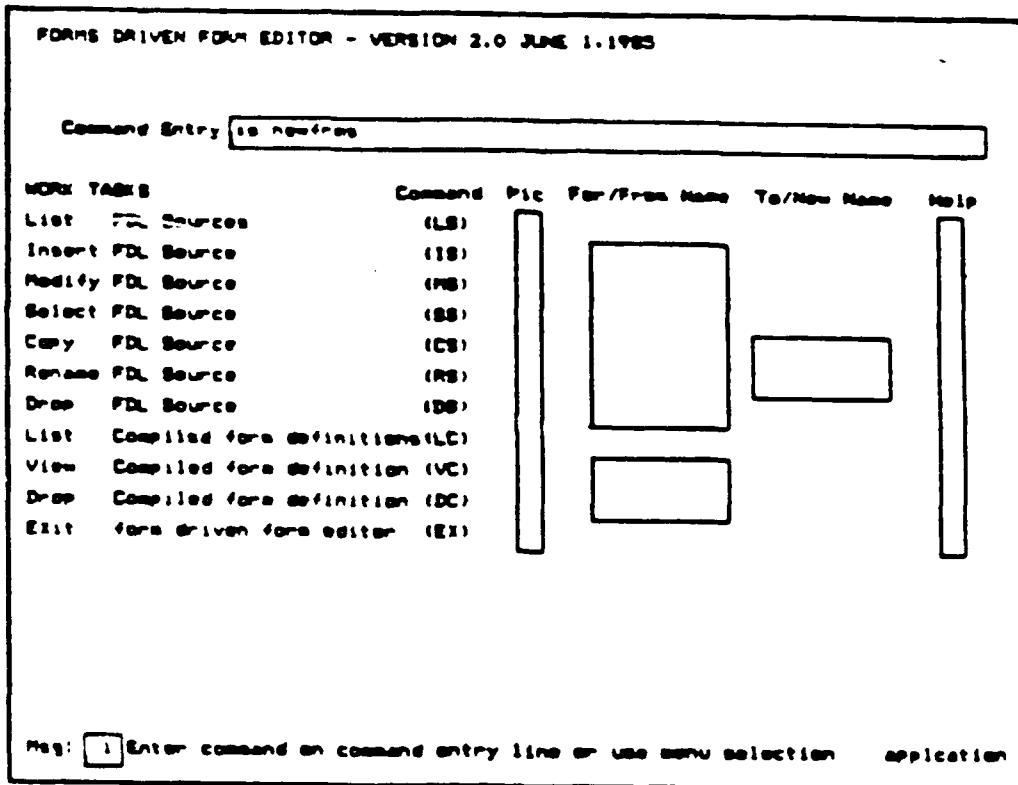


Figure 5-5 Choosing a Work Task-Command Entry
Note that command entry overrides menu selection.

The following sections summarize the functions performed by each of the Work Tasks.

5.4.2 LS (List FDL Sources)

Use this task to display the names of all your FDL source files.

The Command Entry format is: LS<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for List form language Sources and press the <ENTER> key. For example:

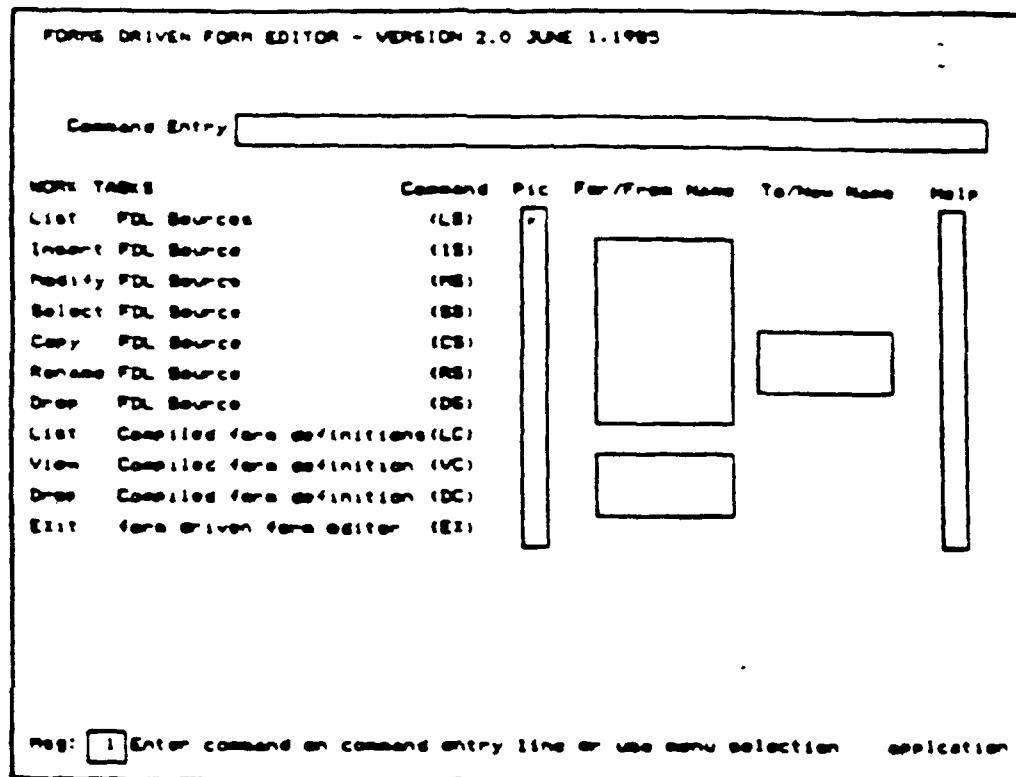


Figure 5-6 Choosing LS Using Menu Selection

When this task is executed, the next screen displayed is:

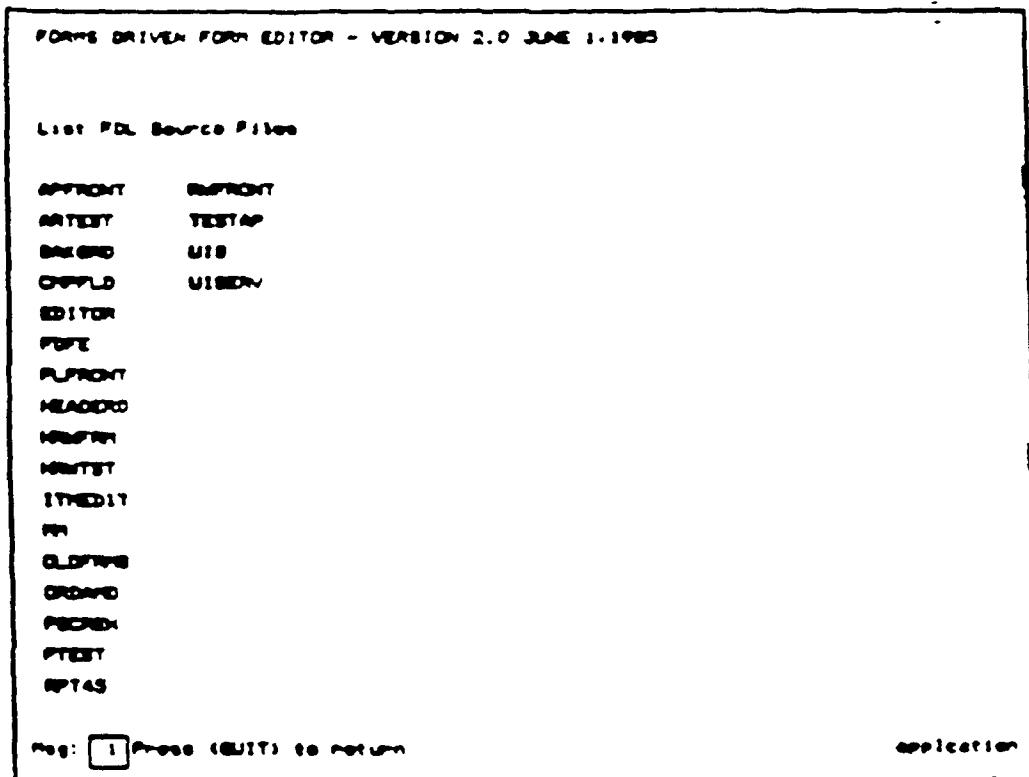


Figure 5-7 List FDL Source Files Screen

5.4.2.1 List FDL Source Files Screen

This screen lists the names of all your FDL source files. When there are more names than can be displayed on the screen, the message "Press <ENTER> for more names or <QUIT> to return" is displayed in the message line. This will continue until all the names have been displayed and then the message "Press <QUIT> to return" is displayed. You will return to the Work Task screen when you press the <QUIT> key.

5.4.3 IS (Insert FDL Source)

Use this task to define new form(s) and store them in a new FDL source file. You must enter the name of the new FDL source file you want to create without an extension or directory name. If the name you enter already exists as an FDL source

file, an error message will be displayed in the message line.

The Command Entry format is: IS filename<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Insert form language Source, the name of the new FDL source file as the "For/From Name", and press the <ENTER> key. For example:

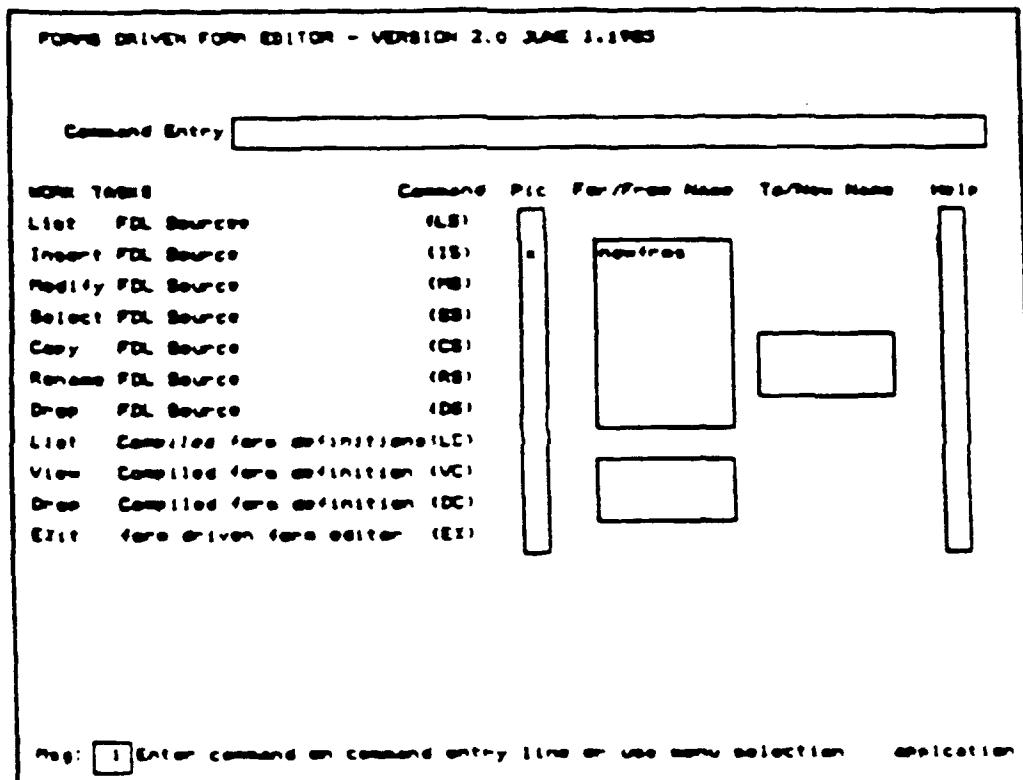


Figure 5-8 Choosing IS Using Manual Selection

When this task is executed, the next screen displayed is the Edit Task screen as shown in section 5.5.

5.4.4 MS (Modify FDL Source)

Use this task to modify an existing FDL source file. This includes inserting new form definitions in the file and

modifying any that the file already contains. You must specify the name of the FDL source file.

The Command Entry format is: MS filename<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Modify form language Source, the existing FDL source file name as the "For/From Name", and press the <ENTER> key. For example:

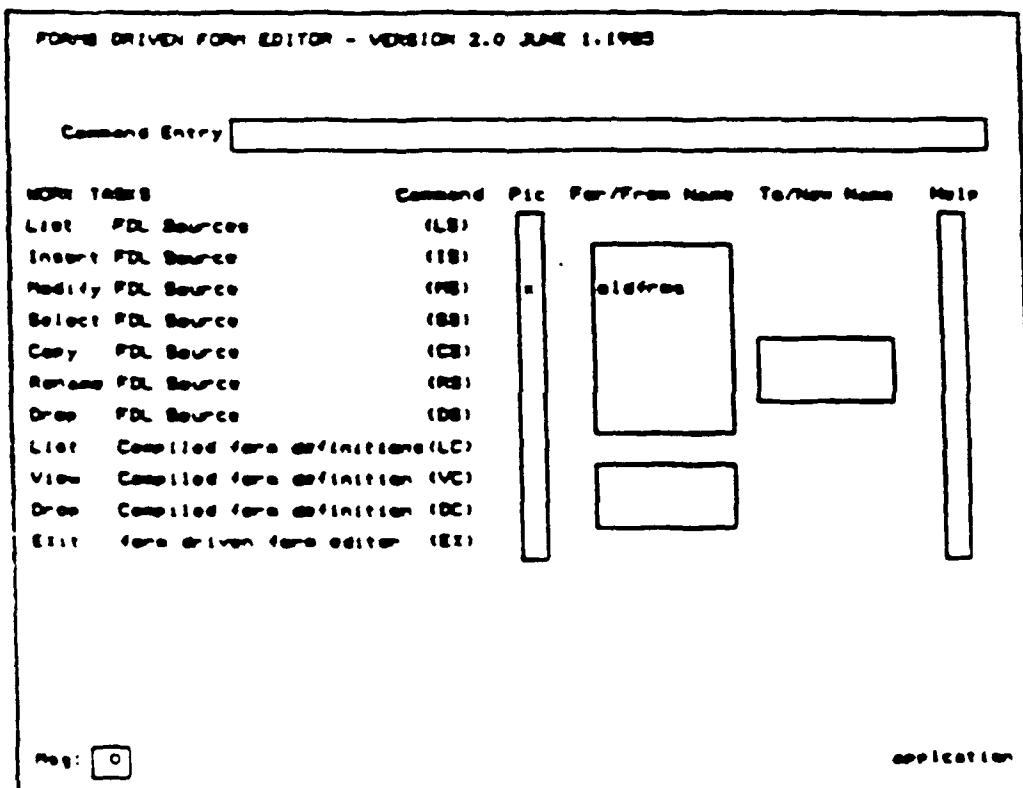


Figure 5-9 Choosing MS Using Menu Selection

When this task is executed, the next screen displayed is the Edit Task Screen as shown in section 5.5.

5.4.5 SS (Select FDL Source)

Use this task to review the form definitions contained in an FDL source file. You must enter the name of the FDL source file that you want to review. Options in this task include listing the names of all the forms defined in this source file, displaying a form as it will appear when used in an application program, and reviewing the characteristics of an individual form in a read only mode.

The Command Entry format is: SS filename<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Select form language Source, the existing FDL source file name as the "For/From Name", and press the <ENTER> key. For example:

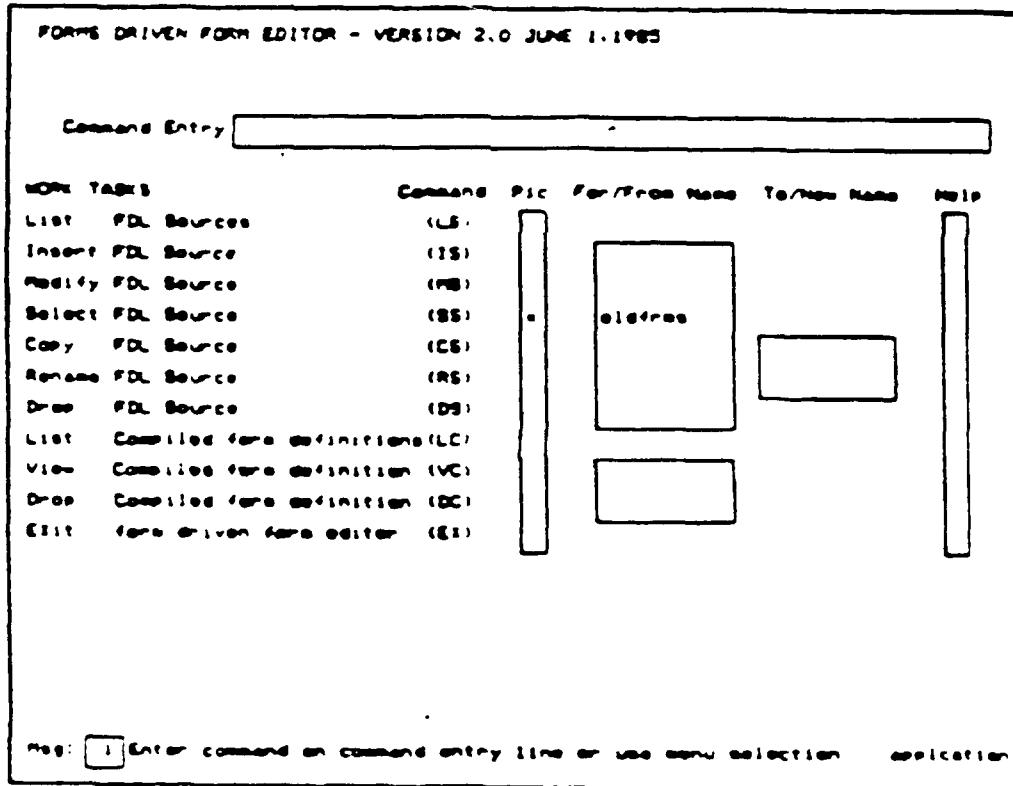


Figure 5-10 Choosing SS Using Menu Selection

When this task is executed, the next screen displayed is a limited Edit Task screen as shown in section 5.6.

5.4.6 CS (Copy FDL Source)

Use this task to make a copy of an existing FDL source file. You must enter the name of the existing file and a new name for the copy.

The Command Entry format is: CS fromname toname<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Copy form language Source, the existing FDL file name as the "For/From Name", the copy file name as the "To/New Name", and press the <ENTER> key. For example:

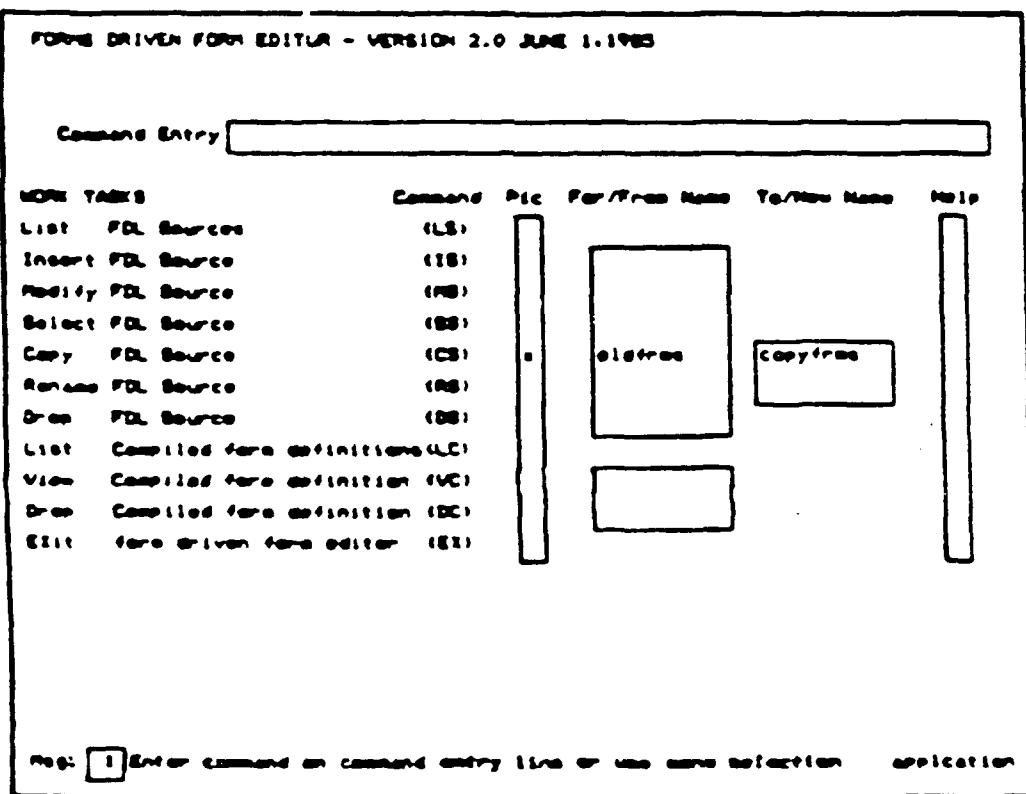


Figure 5-11 Choosing CS Using Menu Selection

When this task is executed, the message "Copy was successful" is displayed in the message line and you remain at the Work Task level.

5.4.7 RS (Rename FDL Source)

Use this task to rename an existing FDL source file. You must enter both the old and the new names for the file.

The Command Entry format is: RS oldname newname<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Rename form language Source, the old file name as the "For/From Name", the new file name as the "To/New Name", and press the <ENTER> key. For example:

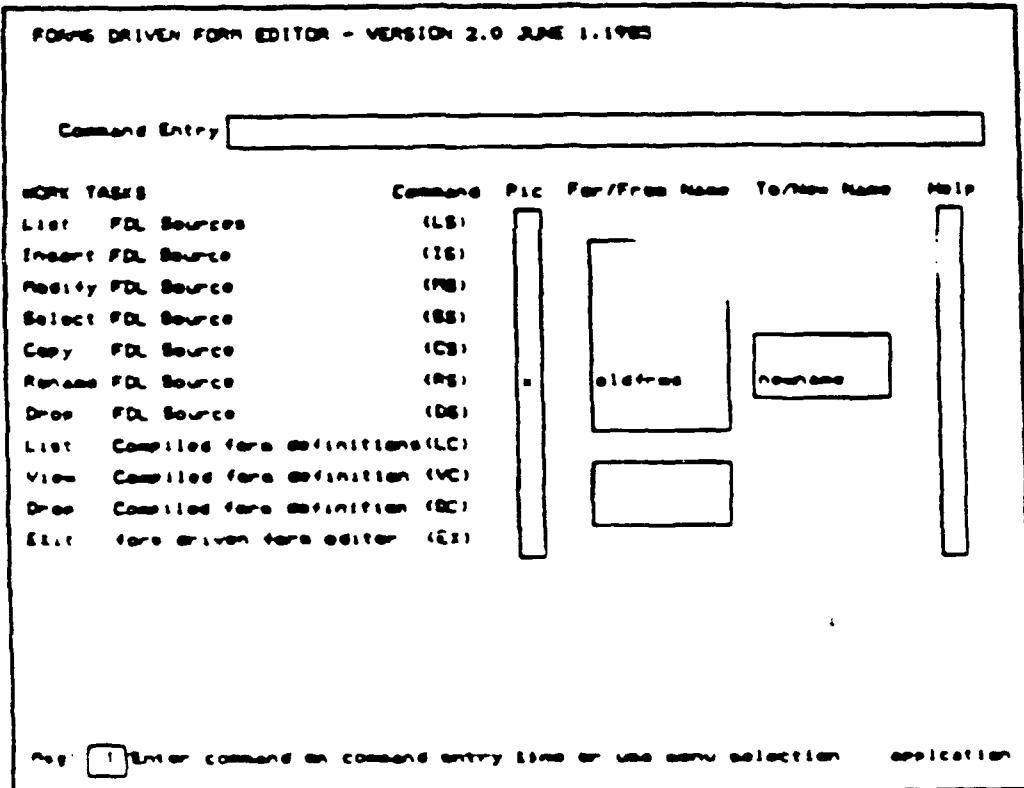


Figure 5-12 Choosing RS Using Menu Selection

When this task is executed, the message "Rename was successful" is displayed in the message line and you remain at the work task level.

5.4.8 DS (Drop FDL Source)

Use this task to drop (delete) an entire FDL source file from your Form Definition Language source files. You must enter the name of the FDL file you want to delete. If this file has been compiled, the DC task must be used to delete the compiled forms that were created from this FDL file.

The Command Entry format is DS filename<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Drop form language Source, the name of the FDL source file you want to delete as the "For/From Name", and press the <ENTER> key. For example:

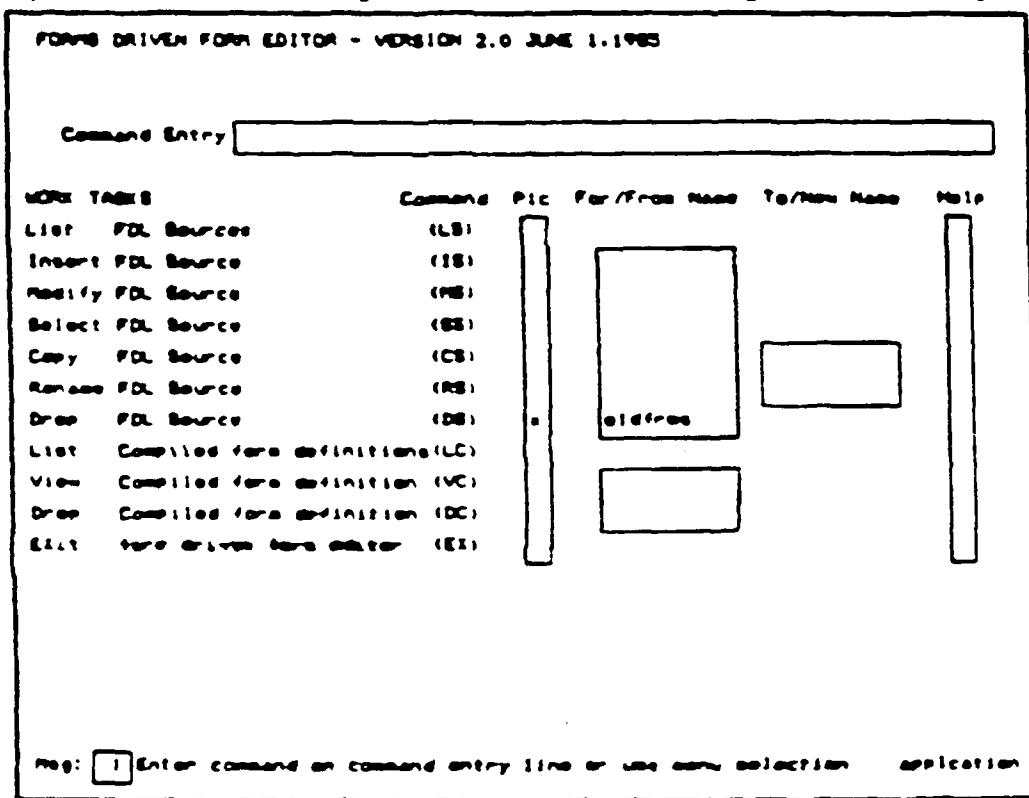


Figure 5-13 Choosing DS Using Menu Selection

When this task is executed, the message "Drop was successful" is displayed in the message line and you remain at the work task level.

5.4.9 LC (List Compiled form definitions)

Use this task to display a list of all your compiled forms.

The Command Entry format is: LC<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for List Compiled form definitions and press the <ENTER> key. For example:

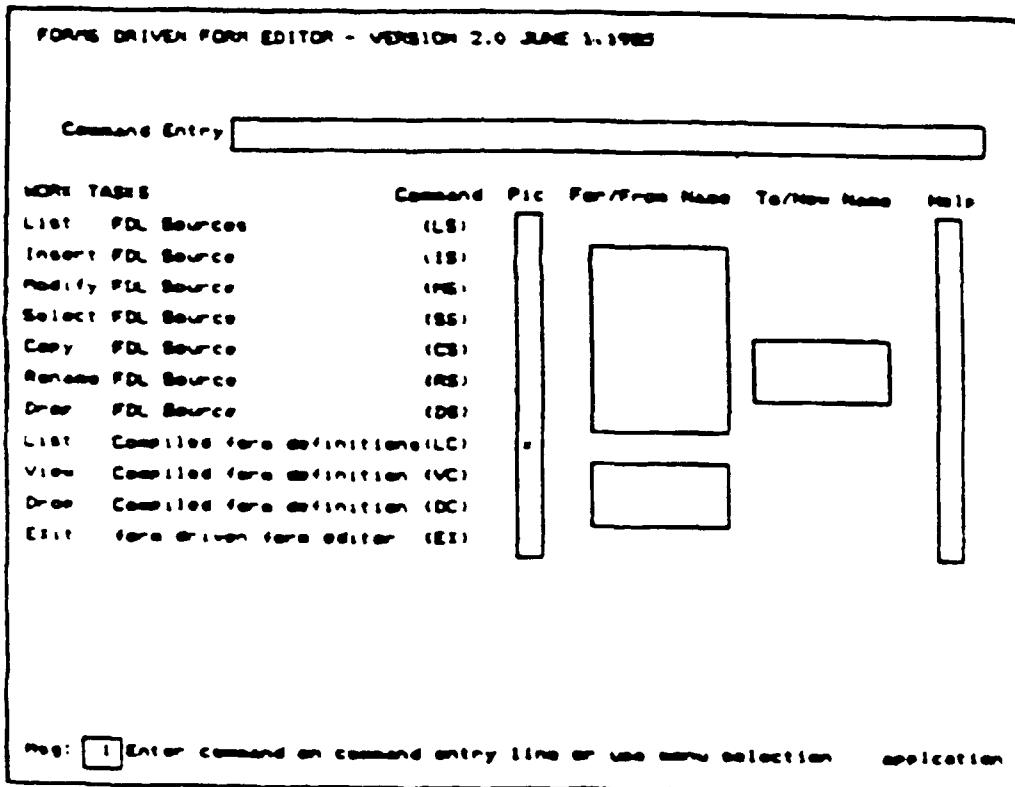


Figure 5-14 Choosing LC Using Menu Selection

When this task is executed, the next screen displayed is:

FORMS DRIVEN FORM EDITOR - VERSION 2.0 JUNE 1.1985				
List Compiled Form Definitions				
STRINFO2	FD0001	GR0003	FD0007	TYPEHELP
STRINFO3	FD0002	GR0004	FD0008	UPDCOM
STRINFO4	FD0003	GR0005	FD0009	VALHELP
STRINFO5	FD0004	GR0006	FD0010	VCHELP
STRONLY	FD0005	GR0007	FD0011	WHHELP
STRVAL	FD0006	GR0008	FD0012	VALCOM1
STRVAL	FD0007	GR0009	FD0013	VALCOM2
LAYOUT	NAMEFORM	FD0010	FD0014	VALCINF
LHELP	NAMEHELP	FD0011	FD0015	VALCIST
LHELP	GR0001	FD0012	FD0016	VALCINP
UHELP	GR0001A	FD0013	FD0017	VALCER
LADFORM	GR0002	FD0014	FD0018	VALCIST
LHELP	GR0002A	FD0015	FD0019	VALCERU
LHELP	GR0003	FD0016	FD0020	VALCER
RDW	GR0003A	FD0017	FD0021	VALCER
RDHELP	GR0004	FD0018	FD0022	VALCER
RD	GR0004A	FD0019	FD0023	VALCER

Page: Press <ENTER> to return

application

Figure 5-15 List Compiled Form Definitions Screen

5.4.9.1 List Compiled Form Definitions Screen

This screen lists the names of all your FD files. When there are more names than can be displayed on the screen, the message "Press <ENTER> for more names or <QUIT> to return" is displayed in the message line. This will continue until all the names have been displayed and then the message "Press <QUIT> to return" is displayed. You will return to the work task screen when you press the <QUIT> key.

5.4.10 VC (View Compiled form definition)

Use this task to display a compiled form as it would appear when used in an application program. You must enter the name of the form you want to display.

The Command Entry format is: VC formname<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for View Compiled form definition, the name of the form as the "For/From Name", and press the <ENTER> key. For example:

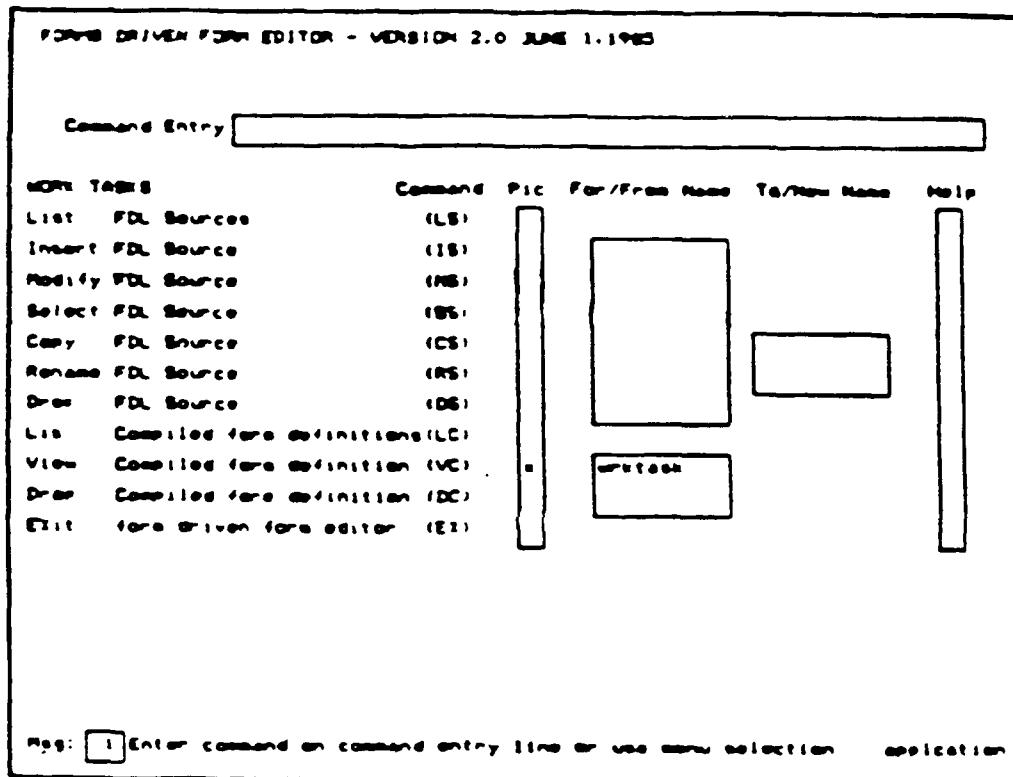


Figure 5-16 Choosing VC Using Menu Selection

When you execute this task, the form is displayed on the screen and the message "User view of the form, press the <QUIT> key to return" is displayed in the message line. You will return to the Work Task screen when you press the <QUIT> key.

5.4.11 DC (Drop Compiled form definition)

Use this task to drop (delete) a compiled form from your compiled forms. You must enter the name of the form you want to delete.

The Command Entry format is: DC formname<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Drop Compiled form definition, the name of the form as the "For/From Name", and press the <ENTER> key. When this task is executed, the message "Drop was successful" is displayed in the message line and you remain at the Work Task level. For example:

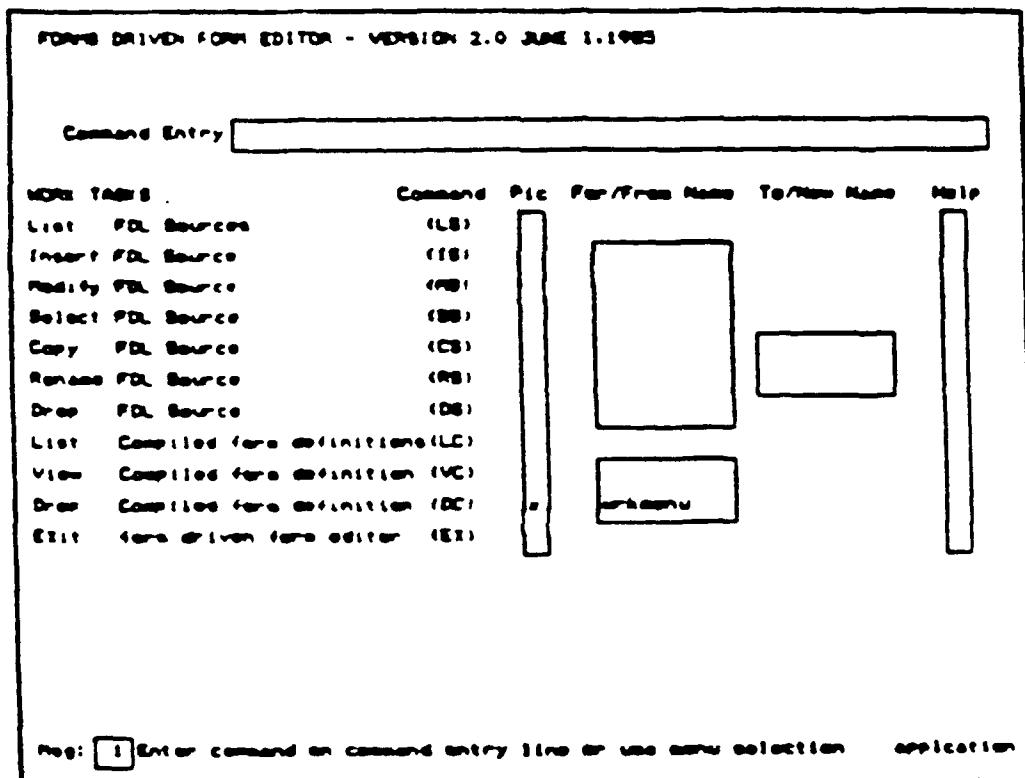


Figure 5-17 Choosing DC Using Menu Selection

5.4.12 EX (EXit form driven form editor)

Use this task to terminate your current FDFE edit session.

The Command Entry format is: EX<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for EXit form driven form editor and press the <ENTER> key. Pressing the <QUIT> key from anywhere on the work task screen is another way to terminate your current FDFE edit session.

5.5 Edit Task Screen

This screen is a list of the FDFE functions that allow you to maintain the form definitions contained in the specified FDL source file and edit a specific form definition. It is the next screen displayed when you execute the IS or MS work task.

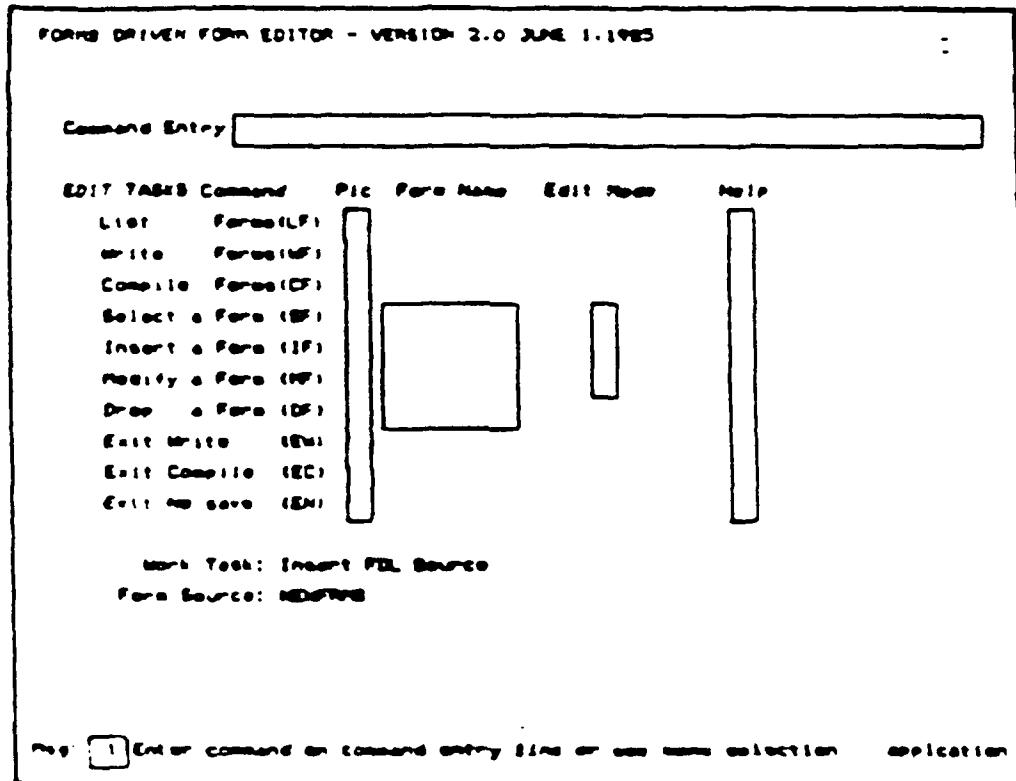


Figure 5-18 Edit Task Screen

Note that the order of the edit tasks on the screen does not imply an execution order. For example, if you choose IS, you are creating a new FDL source file and there won't be any form definitions to list or modify until you have executed the INSERT task.

The values displayed in the "Work Task" and "Form Source" fields tell you which work task you chose and the FDL source file you are working with, respectively.

5.5.1 Choosing an Edit Task

Edit tasks can be chosen using menu selection or command entry as explained in the "Choosing a Work Task" section of this manual.

The following sections explain each of the edit tasks.

5.5.2 LF (List Forms)

Use this task to display the names of all the forms defined in the specified FDL source file. These names correspond to the form_name on the CREATE FORM entries of the FDL syntax as explained in the Form Definition Language Section of this manual.

The Command Entry format is: LF<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for List Forms and press the <ENTER> key. For example:

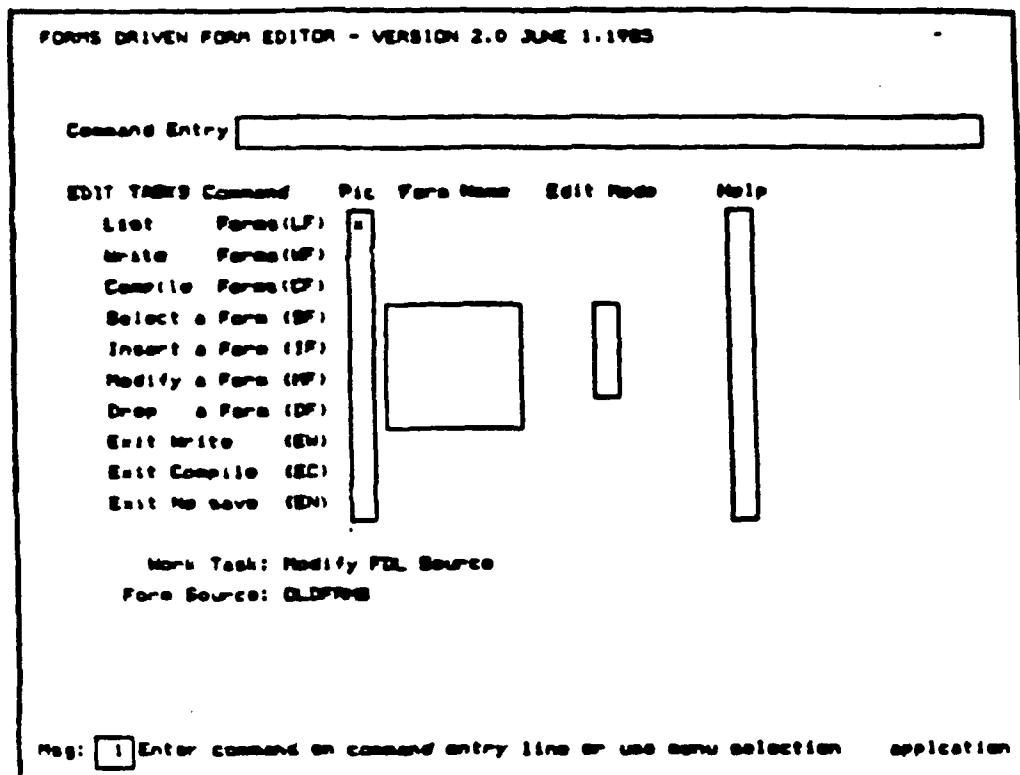


Figure 5-19 Choosing LF Using Menu Selection

When this task is executed, the next screen displayed is:

List of Forms for FDL File: GLOBFORM					
EDITASK	ISTRINGS	CDR2	FDHELP	LDHELP	FDHELP
EDTALL	ISTRINGS2	ITCODES	EDHELP	FTHELP	RDHELP
EDTLINE	ISTRINGS2	PRINFO	LDHELP	FTHELP2	MLHELP
EDTREL	ISTRINGS3	PRINFO	VCHELP	FDHELP	VALHELP
CLINFO	ISTRINGS3	PRINFO	EDHELP	EDHELP	EDHELP
WTASK	WALC01	COMINFO	UHELP	SHHELP	RDHELP
FDTEVEN	ISTRINGS4	FLDARY	SHHELP	FTHELP	FDHELP
WDPMENU	ISTRINGS4	ITHELP	EDHELP	BDHELP	
WALEDIT	WALC02	ITMAIL	WHELP	FDHELP	
WALCINFO	ISTRINGS5	LAYOUT	CHHELP	TYHELP	
DELFDL	ISTRINGS3	EDTLPLS	EDHELP	WHELP	
DELML	ISTRINGS3	MLLIST	EDHELP	FTHELP	
PLSTOP	PLD01	LMHELP	IPHELP	LOCKHELP	
CONTCTL	PLCINFO	ISHELP	WHELP	FDHELP	
PLDINFO	ITDOLY	RDHELP	SHHELP	BDHELP	
RDPLD	COMINFO	SHHELP	SHHELP	BDHELP2	
ISTRFORM	CDR1	CDHELP	SHHELP2	FDHELP	

Key: Press <ENTER> to return application

Figure 5-20 List of Forms Screen

5.5.2.1 List of Forms Screen

This screen lists the names of all the forms defined in the current FDL source file. When there are more names than can be displayed on the screen, the message "Press <ENTER> for more names or <QUIT> to return" is displayed in the message line. This will continue until all the names have been displayed and then the message "Press <QUIT> to return is displayed. You will return to the Edit Task screen when you press the <QUIT> key.

5.5.3 WF (Write Forms)

Use this task to save the form definition(s) you are currently working on in the FDL source file shown in the "Form Source" field.

The Command Entry format is: WF<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Write Forms and press the <ENTER> key. For example:

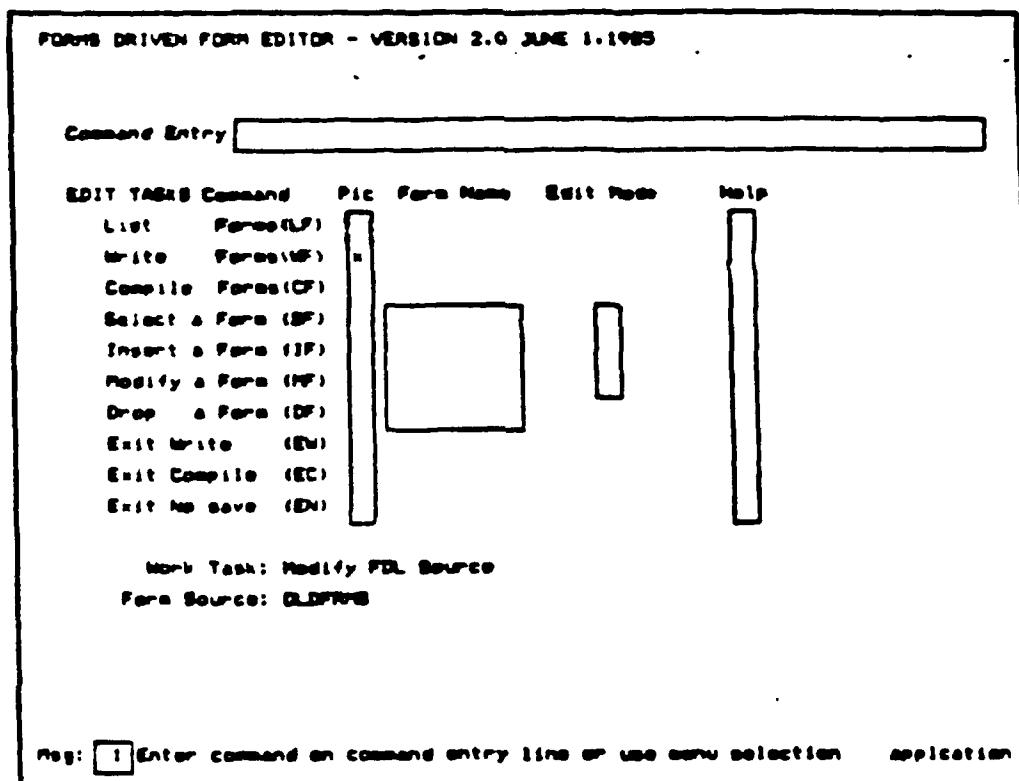


Figure 5-21 Choosing WF Using Menu Selection

When this task has executed, the form definitions are compiled since only syntactically correct form definitions can be edited in the FDFE. Any syntax errors are stored in the message queue and can be displayed using the <MESSAGE QUEUE> key. When there are no syntax errors and the write is complete, you remain at the Edit Task level. Note that no FD files are created, only the FDL file is created.

5.5.4 CF (Compile Forms)

Use this task to compile and save the form definition(s) contained in the FDL source file shown in the "Form Source" field without exiting the FDFE.

The Command Entry format is: CF<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Compile Forms and press the <ENTER> key. For example:

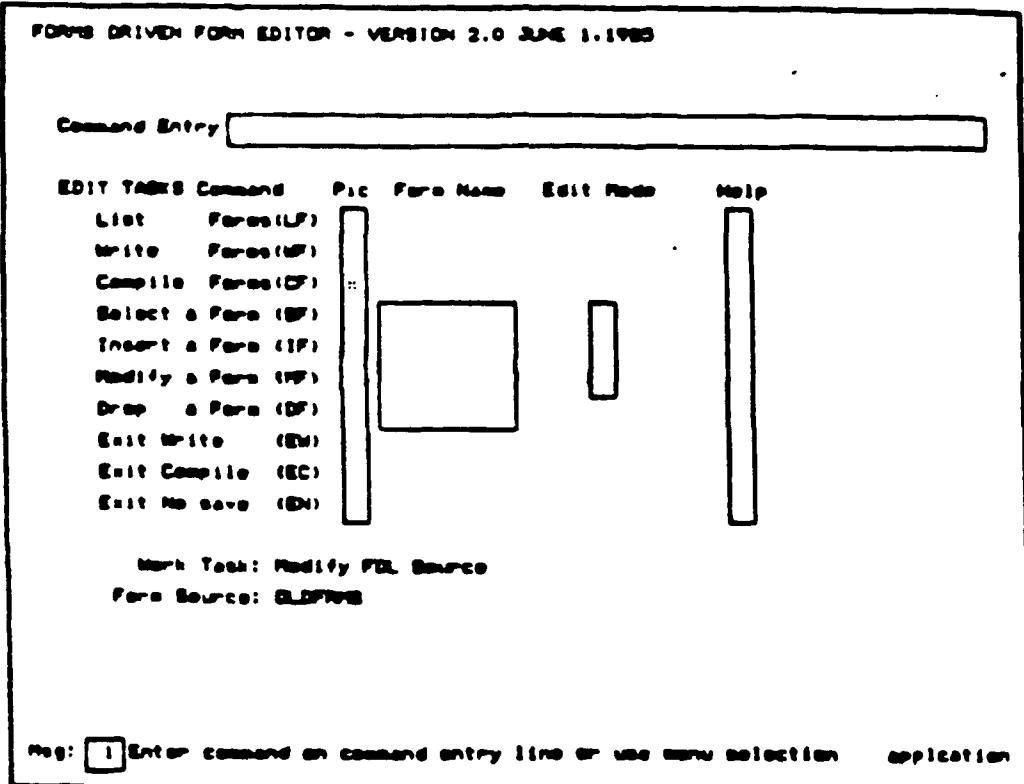


Figure 5-22 Choosing CF Using Menu Selection

If there are syntax errors, the message "Form compile errors, review and correct" is displayed in the message line. The errors are stored in the message queue which can be displayed using the <MESSAGE QUEUE> key. When there are syntax errors, you must correct them and reexecute the CF task in order to obtain both an FDL file and FD file(s) using FDFE tasks. If you exit without correcting the syntax errors, the FDL source file is saved as a TMP file that may be edited using EDT or some other editor and then compiled using FLAN. The FDFE can only modify FDL source files free of error.

When the forms are compiled with no syntax errors, the form definitions are saved in the FDL source file shown in the "Form Source" field and the compiled forms are saved as FD files. At this time you can press the <QUIT> key to return to the Work Task screen where you can display a form definition using the VC task or choose another FDL source file to work with.

5.5.5 SF (Select a Form)

Use this task to review the characteristics and attributes of an individual form contained in the FDL source file shown in the "Form Source" field. You must enter the name of the form you want to review and an edit mode to determine the format the form definition will be presented in. Your format choices are S for single field mode, F for form mode, or L for layout mode.

The Command Entry format is: SF formname editmode<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Select a Form, the name of the form definition you want to review as the "Form Name", S,F, or L as the "Edit Mode", and press the <ENTER> key. For example:

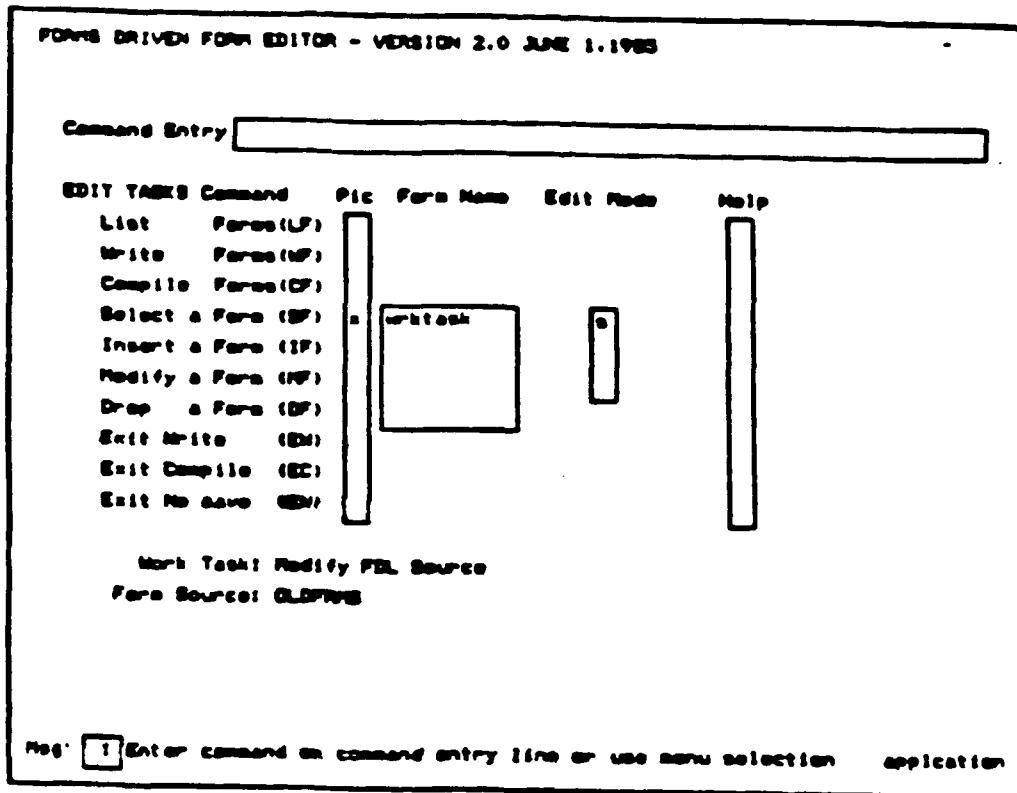


Figure 5-23 Choosing SF Using Menu Selection

When this task is executed, the next screen displayed depends on the Edit Mode you chose. The next three sections describe how to review a form using each of the modes. After reviewing a form, press the <QUIT> key to return to the Edit Task screen.

5.5.5.1 Using Single Field Mode to Review a Form

When you execute the SF edit task with the "S" edit mode, the next screen displayed is:

FIELD EDIT MODE For FDL File Editors

Task S

Field Type Direct Get FDL Form

Form Structure
Size 70 by 17
Background BLACK
Prompt WORK TASKS
Row 6 Column 2

REQUIRED:

FIELD Row Column By

OPTIONAL:

Display Actual Direction Spacing

Field Repetition (→)

Prompt Row Col

ITEM ONLY:

Help Value

Justify Case Data Type Enter/Fill MIN Value MAX Value

Msg: Enter field to be viewed

application

Figure 5-24 Reviewing a Form in Single Field Mode

The SF task only allows you to read the form definition. This is shown by the "S" value in the "TASK" field which cannot be changed. The characteristics that apply to the entire form are displayed in the FORM area of the screen. The characteristics of the individual fields are displayed in the field information template. An explanation of each of the fields in the template is given in sections 5.5.6.1.2, 5.5.6.1.3, and 5.5.6.1.4.

There are several ways to specify which field you want to display. The <NEXT> and <PREV> function keys display the next or previous field with respect to the current field being displayed. Note that when this screen is displayed after choosing the SF edit task, the field information template is blank so you would use the <NEXT> key first.

You can use the "Type" and "Direct" fields in the TASK area of the screen to specify the field to be displayed. Use the "Type" field to say whether you want to display only items, forms, windows, or all the field types. The values are I, F, W, and A. Use the "Direct" field to display the next (N), previous (P), beginning (B) or ending (E) field in the form definition. Note that the "Field" field must be blank when using these fields and the "Type" field is ignored if you use the <NEXT> and <PREV> function keys. After entering the appropriate values, press the <ENTER> key to display the desired field.

You can also display a specific field by entering the field name in the "Field" field in the TASK area of the screen and pressing the <ENTER> key. Note that a value in this field overrides the "Type" and "Direct" fields.

When you are through reviewing the form, press the <QUIT> key to return to the Edit Task screen.

5.5.5.2 Using Form Mode to Review a Form

When you execute the SF edit task with the "F" edit mode, the next screen displayed is:

FORM EDIT MODE		For FDL File CLEPRMS	
TASK S		FORM WORKAREA	
Type	<input type="text"/>	Size	78 by 17
Get FDL	<input type="text"/>	Background	BLACK
	Form	Prompt	WORK TASKS
	<input type="text"/>	Row	6
	<input type="text"/>	Column	2
-- Field Characteristics Table --			
Msg: <input type="text"/> Enter type of fields to be viewed		APPLICATION	

Figure 5-25 Reviewing a Form in Form Mode

The characteristics that apply to the entire form are displayed in the FORM area of the screen. As in Single Field mode, the "TASK" field value is "S" and cannot be changed. The characteristics for the fields defined on the form can be displayed in the FIELD CHARACTERISTICS TABLE. The field characteristics contained in this table correspond to the field information template as described in sections 5.5.6.1.2, 5.5.6.1.3, and 5.5.6.1.4. Use the "Type" field in the TASK area of the screen to specify which fields you want to review. Your choices for this field are:

A to display all the fields on the form.
I to display only the item fields on the form.
F to display only the form fields on the form.
W to display only the window fields on the form.

After entering the "Type" field value, press the <ENTER> key to display the FIELD CHARACTERISTICS TABLE on the screen. For example:

Form Edit Mode										For FDL File CLOPRES		
Task 8										Form Structure		
Type	A	Get FDL	Form	Size			70 by 17	Background	BLACK	Prompt	MORE TASKS	
			<th>Row</th> <td>6</td> <th>Column</th> <td>2</td> <th></th> <th></th> <th></th> <th></th> <th></th>	Row	6	Column	2					
-- Field Characteristics Table --												
Field Name	T	Row	Col	Size	Display	Dsp Act	D Sp	Promt	Pos			
FDIEVEN	F	1	1	70	1	BLACK						
CDI	I	4	19	60	1	INPUT						
WIRMENU	F	7	39	38	1	BLACK						
LHELP	I	7	76	1	1	INPUT						
THHELP	I	8	76	1	1	INPUT						
MHELP	I	9	76	1	1	INPUT						
SHHELP	I	10	76	1	1	INPUT						
CHHELP	I	11	76	1	1	INPUT						
RHELP	I	12	76	1	1	INPUT						
DHELP	I	13	76	1	1	INPUT						
LHELP	I	14	76	1	1	INPUT						
VHELP	I	15	76	1	1	INPUT						
More--												
Page: 0												
application												

Figure 5-26 FIELD CHARACTERISTICS TABLE - Part One

For item fields, the FIELD CHARACTERISTICS TABLE contains more information which you can display by entering any nonblank character in the "More" field below the table on this screen and the next screens displayed and pressing the <ENTER> key. The remainder of the table is shown in the following three screens:

FORM EDIT MODE

For FDL File CLDTYPE

TASK 8

Type

Set FDL Form

FORM WRITE MODE

Size by

Background

Prompt

Row Column

-- Field Characteristics Table --

Item name Help Message

FDTEVEN

CDI

MRKMENU

LHELP FORMLHELP

IHELP FORMIHELP

MHELP FORMMHELP

BHELP FORMBHELP

CHELP FORMCHELP

RHELP FORMRHELP

DHELP FORMDHELP

LHELP FORMLHELP

VHELP FORMVHELP

More--

Page: Application

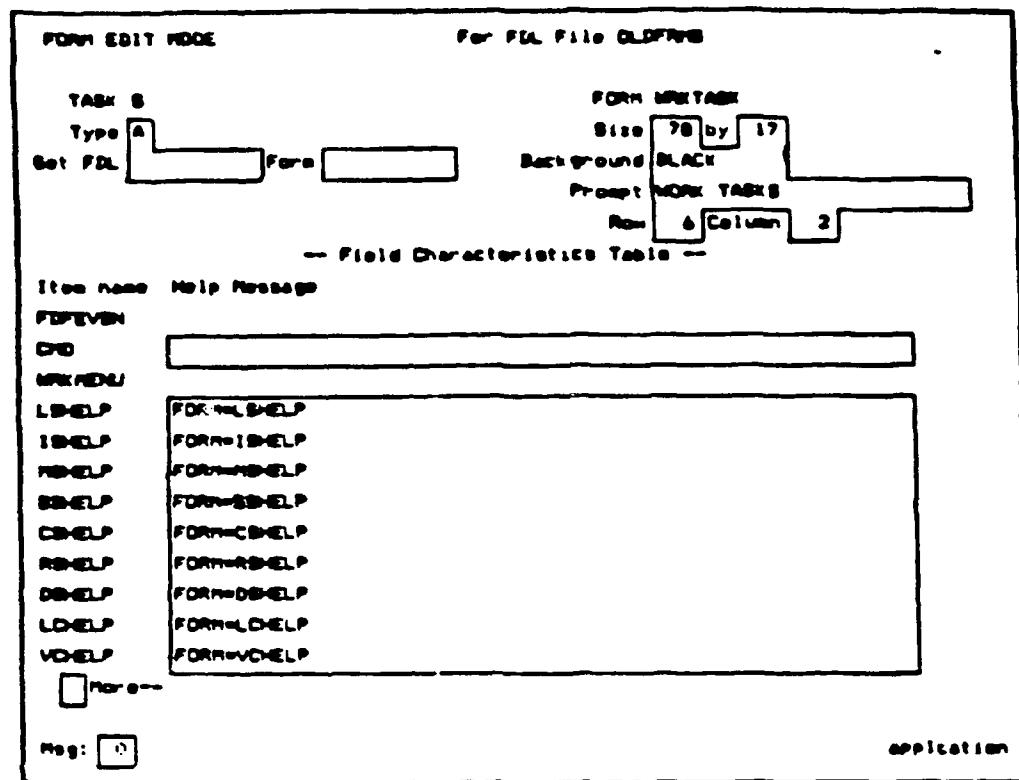


Figure 5-27 FIELD CHARACTERISTICS TABLE - Part Two

FORM EDIT MODE For FDL File OLDTYPE

TABLE 8

Type Form

Get FDL Form

FORM METADATA

Size 78 by 17

Background BLACK

Prompt WORK TASKS

Row 6 Column 2

-- Field Characteristics Table --

Item Name	Item Value
PPFENV	
CDP	
MRMENV	
LWHELP	
ISHELP	
HSHELP	
BSHELP	
CSHELP	
RSHELP	
DSHELP	
LSHELP	
VSHELP	
<input type="checkbox"/> Normal	

Prog: 0

application

Figure 5-28 FIELD CHARACTERISTICS TABLE - Part Three

Figure 5-29 FIELD CHARACTERISTICS TABLE - Part Four

Note that all field names are displayed on these table parts, but values for the characteristics appear for item fields only.

All four parts of the FIELD CHARACTERISTICS TABLE scroll vertically so that you can review all the fields contained in a form if there are more than 12. To activate the paging and scrolling keys, press the <MODE> key until "scroll/page" appears in place of "application" in the mode field as explained in the IISS Terminal Operator Guide. You can also use these keys to scroll the "Prompt" field on the first part of the table horizontally so that an entire prompt can be seen if it contains more than 15 characters. The Field Repetition fields (Dsp, Act, D, Sp) also scroll horizontally. Return the FP mode field to "application" before pressing the <QUIT> key to return to the Edit Task screen.

5.5.5.3 Using Layout Mode to Review a Form

When you execute the SF edit task with the "L" edit mode, the form is displayed using the layout symbols described in section 5.5.6.3. For example:

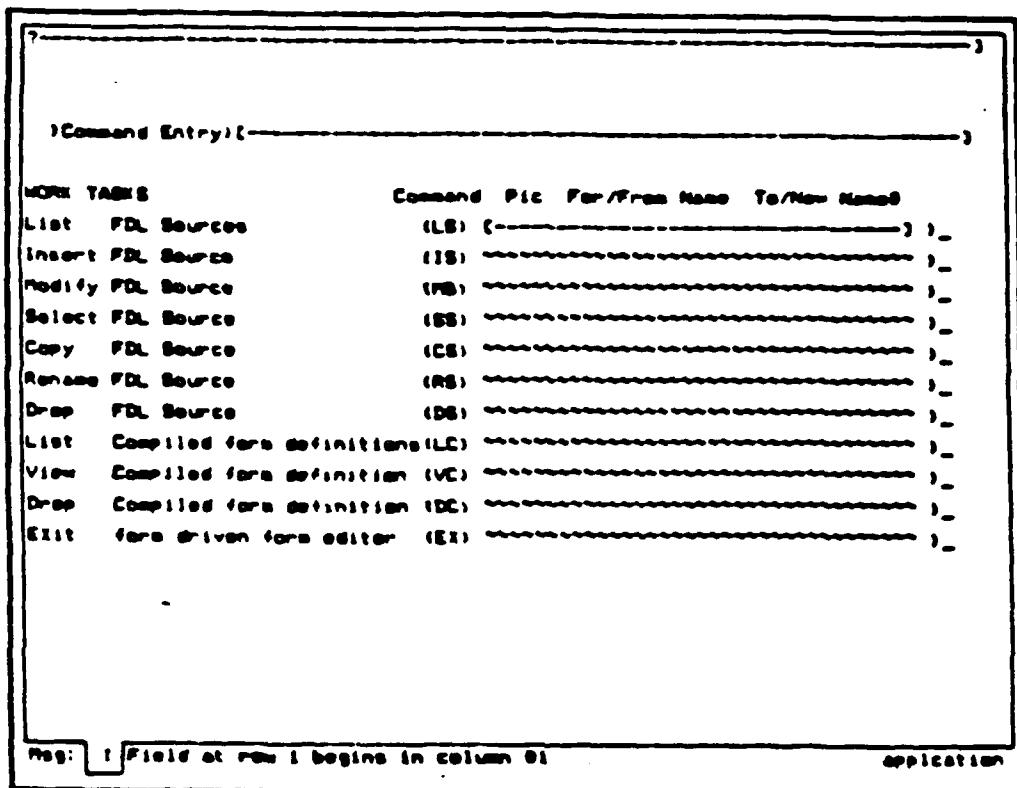


Figure 5-30 Reviewing a Form in Layout Mode

Press the <QUIT> key to return to the Edit Task screen.

5.5.6 IF (Insert a Form)

Use this task to define a new form. When saved, this form definition will be inserted into the FDL source file displayed in the "Form Source" field. You must enter the name of the form that you are going to define and an edit mode to determine the editing style you want to use to define the form.

The Command Entry format is: IF formname editmode<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Insert a Form, the name of the form as the "Form Name", S, F, or L as the "Edit Mode", and press the <ENTER> key. For example:

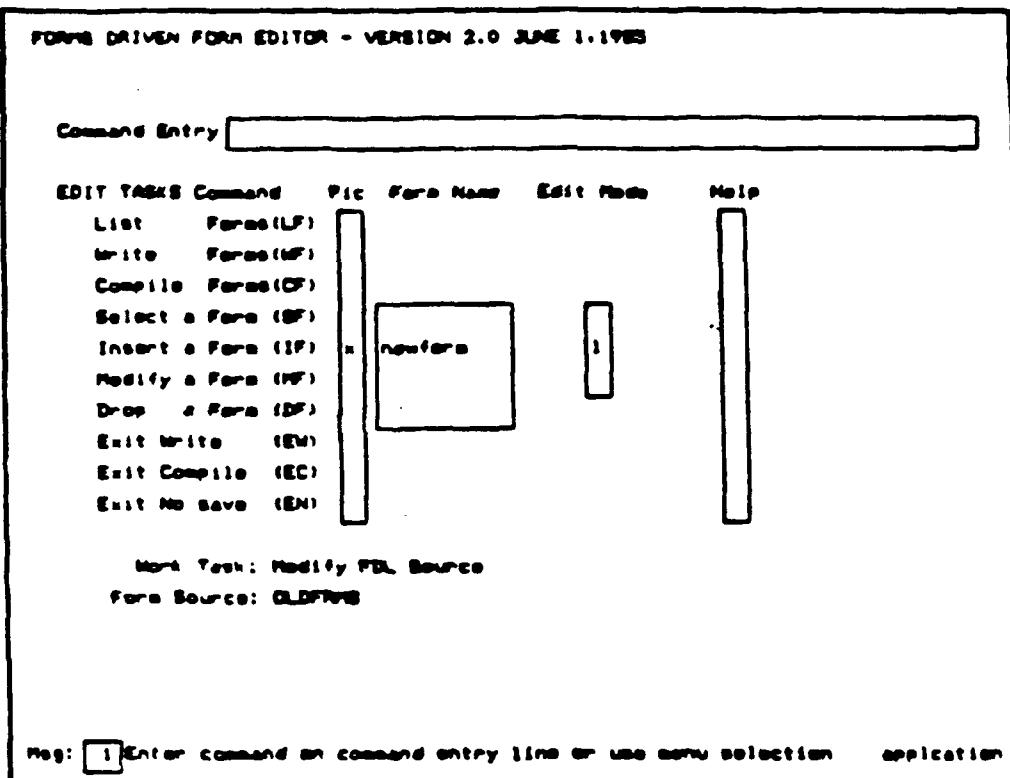


Figure 5-31 Choosing IF Using Menu Selection

The next screen displayed depends on the Edit Mode you choose. The next three sections explain how to define a form using the single field, form, and layout edit modes.

5.5.6.1 Using Single Field Edit Mode to Define a Form

When you execute the IF edit task with the "S" edit mode, the next screen displayed is:

FIELD EDIT MODE		For FDL File Changes	
Task	<input type="text" value="T"/>	Form Newform	
Field	<input type="text" value="F1"/>	Size	0 by 0
Type	<input type="text" value="Text"/>	Background	BLACK
Direct	<input type="checkbox"/>	Prompt	
Get FDL	<input type="checkbox"/>	Row	<input type="text" value="Row"/>
	<input type="checkbox"/>	Column	<input type="text" value="Column"/>
— Field Information —			
REQUIRED:		OPTIONAL:	
FIELD	<input type="text" value="C1"/>	Type	<input type="text" value="Text"/>
Row	<input type="text" value="1"/>	Actual	<input type="checkbox"/>
Size	<input type="text" value="10 by 1"/>	Direction	<input type="checkbox"/>
Display/	<input type="checkbox"/>	Spacing	<input type="checkbox"/>
Background	<input type="checkbox"/>	Prompt	<input type="text" value=""/>
	<input type="checkbox"/>	Row	<input type="text" value="Row"/>
	<input type="checkbox"/>	Col	<input type="text" value="Col"/>
ITEM ONLY:			
Help	<input type="text" value=""/>		
Value	<input type="text" value=""/>		
Justify	<input type="checkbox"/>	Data Type	<input type="text" value=""/>
Case	<input type="checkbox"/>	Enter/Fill	<input type="checkbox"/>
	<input type="checkbox"/>	MIN Value	<input type="text" value=""/>
	<input type="checkbox"/>	MAX Value	<input type="text" value=""/>
Flag: <input type="checkbox"/> Enter task and field to be acted on			
APPLICATION			

Figure 5-32 Defining a Form in Single Field Mode

When defining a new form in single field edit mode, the first thing you need to do is define the attributes that apply to the entire form in the "FORM" area fields. The name of the form that you are defining is displayed as the "FORM" field value. This is the value you entered in the "Form Name" field on the previous screen and it cannot be changed.

5.5.6.1.1 Form Area Fields

Size is used to specify the number of columns and rows the form will occupy when it is displayed. These values are optional and if specified, the size of the form or window this form is displayed in takes precedence. This means that the form will be "clipped" if the form size is larger than the size of the form or window this form is displayed in.

When the form size is smaller than the form window size, the form "grows" to fill the window. If you do not specify the size of the form, it defaults to the size of the form or window it is displayed in. The format is "n by m" where n is the number of columns wide and m is the number of rows deep.

Background allows you to define the background of the form. This is analogous to printing paper forms on different colors of paper. Your options are BLACK, WHITE, RED, GREEN, YELLOW, BLUE, MAGENTA, and CYAN. The color of the characters will either be black or white depending on the contrast. NOTE that an application that uses forms with colors other than black and white can still be run on a terminal that does not support colors. In this case the forms will be displayed either normal (white characters on a black background) or reverse (black characters on a white background) depending on the brightness of the background color versus the character color. This field defaults to XPARNT if left black. This means that the form inherits the background of the window or form it is displayed in.

Prompt is used to display text strings on the form that are not related to any given field. This type of prompt is used primarily for titles and formatting. This field scrolls horizontally to allow the text strings to be up to 75 characters long and vertically so that up to 30 form prompts can be defined. To activate the paging and scrolling keys, press the <MODE> key until "scrl1/page" appears in place of "applcation" in the mode field. When you are through defining the form prompts, press the <MODE> key again until "applcation" appears in the mode field.

Row and Col are used to specify where the first character of a form prompt will be positioned on the form. The prompt is positioned with respect to the upper left corner of the form. When the "Prompt" field is scrolled vertically, these fields are also scrolled so that the position of each form prompt can be specified.

You define the fields for the form one at a time using a field information template. The INSERT task allows you to enter the desired values in the template. The COPY task retrieves the values from fields defined on an existing form. You can then modify these values and execute the INSERT task. The next three sections explain the information that defines a field.

5.5.6.1.2 Required Field Information

FIELD is a unique name of up to 10 letters, numbers, and/or underscores that identifies the current field you are defining. Note that if the field is a form, underscores may not be used.

Type (T) is used to identify the field you are defining as a form (F), window (W), or item (I).

Row and Col are used to specify where the field will be positioned on the form. The field is positioned with respect to the upper left corner of the form.

Size is used to determine the area on the form that each occurrence of the field will occupy. The format is "n by m" where n is the number of columns wide the field is and m is the number of rows deep the field is. Note that when a form is displayed in a form or window field, the size of the form or window takes precedence over the defined size of the form being displayed. This means that the form will be "clipped" if the form definition size is larger than the field size. When the form size is smaller than the field size, the form will "grow" to fill the window.

Display/ Background controls access to an item field and determines how it appears to the user for item fields and is analogous to printing paper forms on different colors of paper for form and window fields. Valid values are:

INPUT which means that the user may enter a value for this item and the value is echoed on the screen. The area where the user may type is highlighted on the form. The IISS Terminal Operator Guide explains what "highlight" means for each terminal type.

OUTPUT which means that only the application program may enter a value for this item. The value is displayed in bold type on terminals that support it.

CALCULATED which is the same as OUTPUT except that only the Form Processor can write to it.

TEXT which is the same as OUTPUT except the value is not displayed in bold type.

TABFLD which is the same as TEXT except that the user can tab to it.

HIDDEN which means that the user may enter a value for this item but the value is not echoed on the screen. This option is typically used for items of privileged information such as passwords. The area where the user may type is highlighted on the form. The IISS Terminal Operator Guide explains what "highlight" means for each terminal type.

BLACK which displays white characters on a black background.

WHITE which displays black characters on a whitebackground (sometimes known as reverse video).

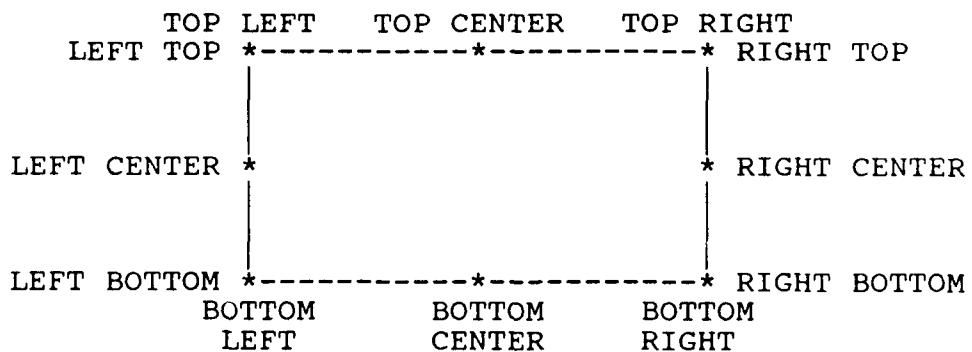
5.5.6.1.3 Optional Field Information

Field Repetition Fields

are used to specify that the field appears on the form more than once to create arrays of fields and whether or not the array is scrollable. The "<->" symbol indicates that these fields all scroll horizontally to allow for multiple array dimensions as explained in the Repeat Spec Entry Item section of this manual. To activate the paging and scrolling keys, press the <MODE> key until "scrl/page" appears in place of "applcation" in the mode field. When you are through defining the field repetition, press the <MODE> key again until "applcation" appears in the mode field. "Display" is how many times to repeat the field on the form. "Actual" is the actual number of times the field occurs. "Direction" tells whether the field repeats in a horizontal (H) or vertical (V) direction. "Spacing" tells how many blank spaces to leave between repetitions.

Prompt is used to specify information such as labels and instructions that will be associated with the field. This field scrolls horizontally so that the prompt text can be up to 75 characters long. The prompt is positioned relative to the field based on the value you enter in the "Pos" field. The paging scrolling keys are activated as explained for the Field Repetition fields.

Pos is used to position a field prompt in one of 12 locations relative to the field. The absolute row and column position is then calculated for you and displayed in the "Row" and "Col" fields after the "Pos" field. The 12 relative locations are:



For prompts to the left of the field, the last character of the prompt is positioned one space left of the field. For prompts to the right of the

field, the beginning character is one space to the right of the field. For prompts above and below the field, care is taken to have the prompt for left begin at the left margin of the field, for right to end at the right margin, and for center to be centered on the width of the field.

5.5.6.1.4 Item Only Information

Help is used to specify a string or another form to provide information that will be displayed when the cursor is in this field and the <HELP> key is pressed. The help string can be up to 60 characters and is displayed in the message line of the screen when the <HELP> key is pressed. For a help form, enter "FORM=formname" where "formname" is the name of the form that you want displayed when the <HELP> key is pressed. No double quotes should be included when entering this information in the "Help" field. The help form can be defined in the current FDL source file or a different one. You can also enter the word "APPLICATION" in the "Help" field to specify that the application program will decide what to do when the user presses the <HELP> key. Quotes are currently not used to enclose a help message but in the next release, standard FDL syntax for the help information will be used. It will no longer be necessary to proceed help form names with "FORM=".

Value is used to specify a default value for an item field. This value will be shown in the field when the form is first displayed to the user. If the item is input and the user does not enter another value, this value is returned to the application program as the value for the item. The default value must be less than or equal to the total number of characters specified by the field size and must be enclosed in double quotes ("default_value"). NOTE that the FDFE only supports string expressions as default values. If a form containing other VALUE expression types is edited using the FDFE, these are enclosed in double quotes thereby converting the expressions to strings.

Justify is used to right (R) or left (L) justify the value in the field.

Data Type is used to specify that the value entered for the field must be numeric (N) or character (C or blank)

MIN/MAX Value is used to set limits for the values that can be entered for numeric item fields. You can define a

range of acceptable values by specifying both a MIN and a MAX value. The field is automatically numeric when a MIN/MAX value is entered.

Case is used to specify that the value entered for the field be converted to upper (U) or lower (L) case.

Enter/Fill is used to specify whether the user must enter a value for the item and whether the value must fill the field. Valid values are:

blank - entering a value is optional.
E - a value must be entered in order to process the form.
F - If a value is entered, it must fill every position of the field must be entered to process the form.

To execute the COPY task, enter "C" in the "TASK" field, use the "Field" or "Type" and "Direct" fields to identify the first field to copy from the existing form, enter values in the "Get FDL" and "Form" fields to identify the existing form, and press the <ENTER> key. You can then use the <NEXT> and <PREV> function keys or the "Field" or "Type" and "Direct" fields to copy the other fields on the existing form. When you have identified a field you want to insert into the current form definition, execute the INSERT task. Note that the COPY task does not change the existing form definition in any way.

To execute the INSERT task, enter "I" in the "TASK" field and press the <ENTER> key. The message "Make your insertions now" is displayed in the message line. At this time you can enter the desired values for the field information or make any modifications to the information copied from an existing form. When the field is defined exactly as you want, press the <ENTER> key again. The message "Field successfully inserted" will be displayed in the message line. If you decide that you do not want to insert the field currently defined in the template, press the <QUIT> key instead of the <ENTER> key to cancel the current INSERT task.

After you have inserted fields in the form, you can use the SELECT, MODIFY, FORM, and DROP field tasks as described in section 5.5.7.1 to review and modify the current form definition.

5.5.6.2 Using Form Mode to Define a Form

When you execute the IF edit task with the "F" edit mode, the next screen displayed is:

FORM EDIT MODE		For FDL File OLDFORM	
Task	<input type="text"/>	FORM NEWFORM	
Type	<input type="text"/>	Size	<input type="text"/> 0 by <input type="text"/> 0
Get FDL	<input type="text"/>	Background	<input type="text"/> BLACK
	<input type="text"/> Form	Prompt	<input type="text"/> Example Form
		Row 1	<input type="text"/> Column 10
— Field Characteristics Table —			
Msg: <input type="checkbox"/> Enter task and type of fields to be acted on		application	

Figure 5-33 Form Mode Screen

When defining a new form in form edit mode, the first thing you need to do is define the attributes that apply to the entire form in the FORM area fields. The name of the form that you are defining is displayed as the "FORM" field value. This is the value you entered in the "Form Name" field on the previous screen and it cannot be changed on this screen. The remaining fields in the FORM area of the screen are described in section 5.5.6.1.1.

Fields for the form are defined by making entries in the FIELD CHARACTERISTICS TABLE. This table is displayed on the screen by executing the INSERT or COPY field task.

The INSERT task allows you to define fields by entering values for the characteristics directly into the table. The COPY task retrieves field characteristics from an exiting form definition and displays them in the table. You can then modify the table entries and execute the INSERT task when the fields are defined as you want them.

To execute the COPY task, enter "C" in the "TASK" field, "I", "F", "W", or "A" in the "Type" field to copy item, form, window, or all field types, the name of the existing form and the FDL it is contained in, and press the <ENTER> key. The fields identified will then be displayed as entries in the FIELD CHARACTERISTICS TABLE. After modifying the entries so that the fields are defined as you want and blanking out the names of the fields you do not want to insert, press the <ENTER> key and then execute the INSERT task. The blanks will be ignored and all fields in the table without blank field names will be inserted.

To execute the INSERT task, enter "I" in the "TASK" field, "A" in the "Type" field, and press the <ENTER> key. The message "Make your insertions now" is displayed in the message line. At this time you can enter the desired values for the field information or make any further modifications to the field information copied from the existing form. For example:

FORM EDIT MODE

FOR FDL FILE CLOFRMS

Tab: 1 Type: a

Set FDL Form: []

FORM NEWFORM
Size: 0 by 0
Background: BLACK
Prompt: EXAMPLE FORM
Row 1: Column 10

-- Field Characteristics Table --

Field Name	T	Row Col	Size	Display	Def Act	D Sp	Prompt	Pos
FIELD1	I	3 12	12	I	INPUT		Item 1:	LT
FIELD2	I	4 13	15	I	INPUT		Item 2:	LT
FIELD3	M	7 15	15	S	BLACK		Message 1	TL

More? []

Msg: [] Make your insertions now!

application

Figure 5-34 Defining a Form in Form Mode

To define item fields there are three more parts to the table which you display by entering any nonblank character in the "More" field below the first three parts of the table as explained in section 5.5.5.2. All four parts of the table scroll vertically so that up to 30 fields can be defined. To activate the paging and scrolling keys, press the <MODE> key until "scrl1/page" appears in place of "applcation" in the mode field.

The information that defines a field in this table corresponds to the field information template used to define a field in Single Field edit mode as explained in sections 5.5.6.1.2, 5.5.6.1.3, and 5.5.6.1.4.

When the fields are defined exactly as you want, press the <ENTER> key again. A message saying the last field defined in the table was successfully inserted is displayed in the message line. The number preceding the message line tells how many fields were inserted. You can use the <MESSAGE QUEUE> key to view the message queue and verify that the other fields in the table were successfully inserted.

After you have inserted fields on the current form, you can use the SELECT, MODIFY, FORM, and DROP field tasks as explained in section 5.5.7.2 to review and modify them.

5.5.6.3 Using Layout Mode to Define a Form

When you execute the IF edit task with the "L" edit mode, the next screen displayed is:

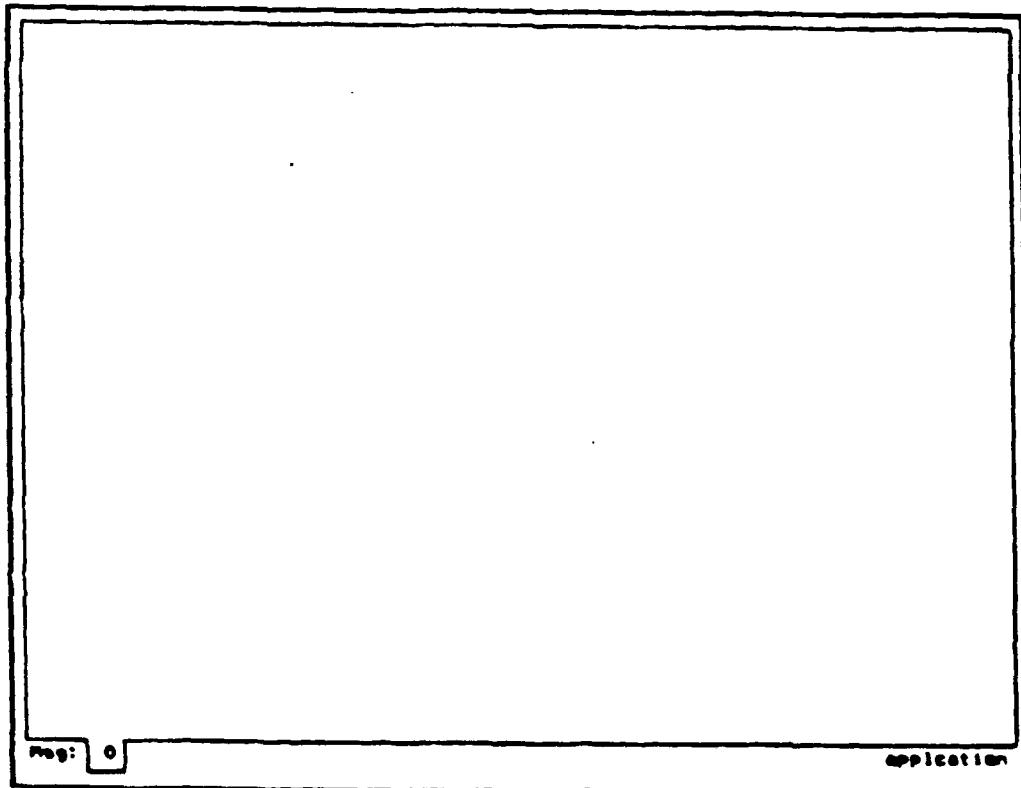


Figure 5-35 Layout Mode Screen

This screen allows you to graphically position fields and prompts on the form using special symbols. This saves time since location parameters and field sizes are calculated for you. However, all fields defined in Layout mode are assumed to be input items and you cannot define arrays of fields. The fields are also assigned default names of the form FLD_R_C where R and C are the starting row and column positions respectively of the field location. To override these defaults, you can press the <SWITCH> key to go into Single Field mode and modify the information for the individual fields. You can specify which field to work on first by positioning the cursor on the first position of the desired field before pressing the <SWITCH> key. You can remain in Single Field mode to modify any of the fields as explained in section 5.5.7.1 or insert new fields as explained in section 5.5.6.1. Pressing the <SWITCH> key again will return you to Layout mode. Whether or not you use the <SWITCH> key, when the form is defined as you want it, press the <ENTER> key to record the form definition. Then press the <QUIT> key to return to the Edit Task screen and execute the WF or EW task to save the form definition. NOTE that if you press the <QUIT> key to return to the Edit Task screen without first pressing the <ENTER> key, you will lose the current form definition data entered.

5.5.6.3.1 Layout Mode Symbols

- Indicates a field position. For example, _____ defines a five position field. Only one dimension horizontal fields may be defined this way. Note that when fields defined this way are reviewed in Layout mode, they are converted to the bracket format described in the next paragraph.
- { } Indicates the first and last positions of a field, inclusively. For example, { } also defines a five position field. These symbols may be used to define two dimension as well as one dimension fields. When displayed in layout mode, the area between the { } is filled with dashes. For example, {----} or

```
{-----  
-----  
-----}.
```
- @ Defines enclosed text as background text on the form.
- > Defines enclosed text as a prompt associated with field to immediate right.
- < Defines enclosed text as a prompt associated with field to immediate left.
- " Defines enclosed text as a prompt associated with field directly below.
- ^ Defines enclosed text as a prompt associated with field directly above.

Note that the starting or ending symbol for a prompt is not necessary if the first or last character of the prompt is at the edge of the form.

- Indicates occurrences of fields that have been defined as arrays using Single Field or Form mode.
- & Indicates a field to be moved when entered in first position of the field. Then position the cursor at new field location and press the <MOVE> key. Note that only one field at a time may be moved.
- ? Indicates an error when in the first position of a field.

The following screen shows how the form "NEWFORM" defined as shown in section 5.5.6.2 appears in Layout mode.

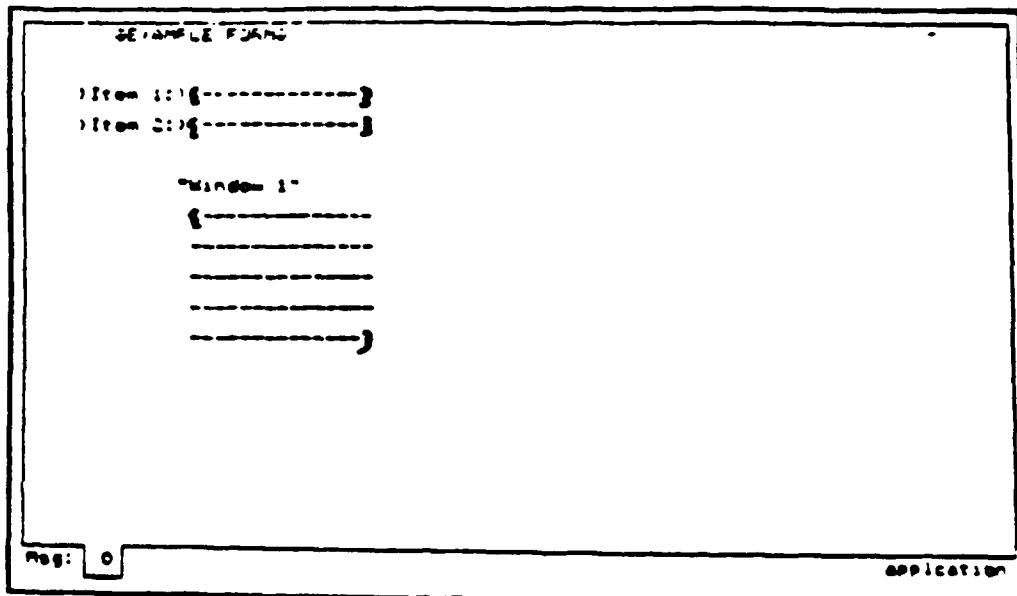


Figure 5-36 Defining a Form in Layout Mode

5.5.6.4 Using Icon Mode to Define a Form

When you execute the IF edit task with the "I" edit mode, the next screen displayed is:

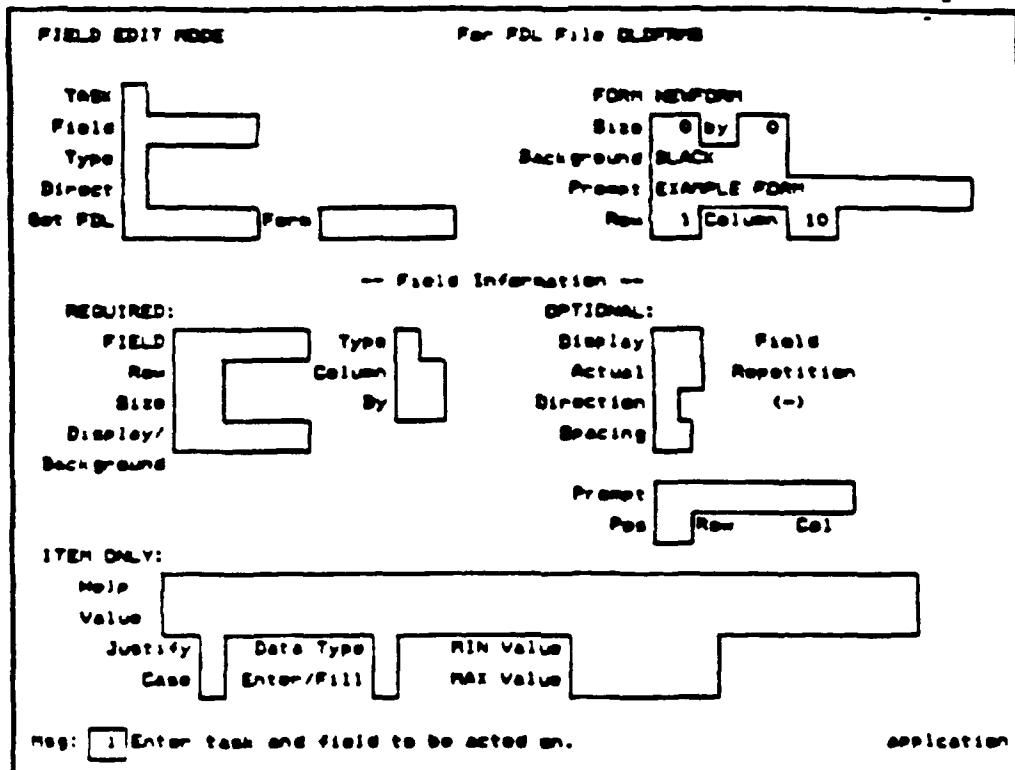
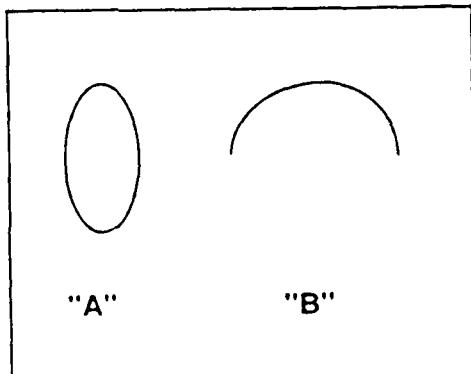


Figure 5-37 Icon Mode Screen

5.5.6.4.1 Creation of an Object

Icons are constructed using multiple objects. To create an object the user may select from four basic styles; circle, arc, box, and polyline. The circle may be stretched to form an ellipse, and then rotated about its center. The arc primitive (a partial circle) may also be stretched and rotated about its center. Note Figure 5-38 demonstrates two such possibilities.

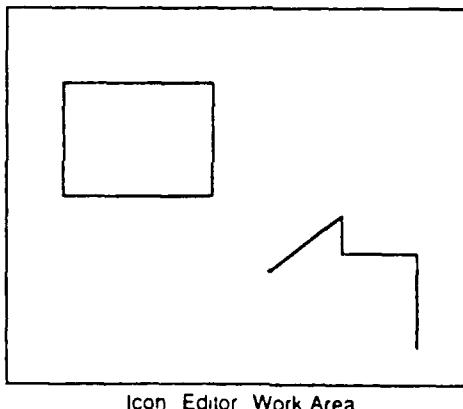
The box can be stretched to make a rectangle in a similiar fashion. The polyline primitive allows the user to create a single line or a series of connected lines. Figure 5-39 shows the box and polyline primitives.



- A- Circle with width greater than depth and rotated about its center.
- B- Arc with width equal to depth no rotation.

Icon Editor Work Area

Figure 5-38 Creation of Circle and Arc Primitives



Icon Editor Work Area

Figure 5-39 Creation of Box and Polyline Pimitives

To select one of the four styles, the user places the cursor on the icon which is to be created and presses <ENTER>.

To make an icon using the polyline primitive requires at least two coordinate points to fully describe a line. Three or more coordinates will make a series of connected lines.

See also Figure 5-40 for an example:

- o the cursor to be placed on the first coordinate with a <PF16> after this placement
- o cursor placement for the second coordinate is selected similarly

- o the last coordinate for the object is selected by positioning the cursor on the desired coordinate and then pressing <ENTER>

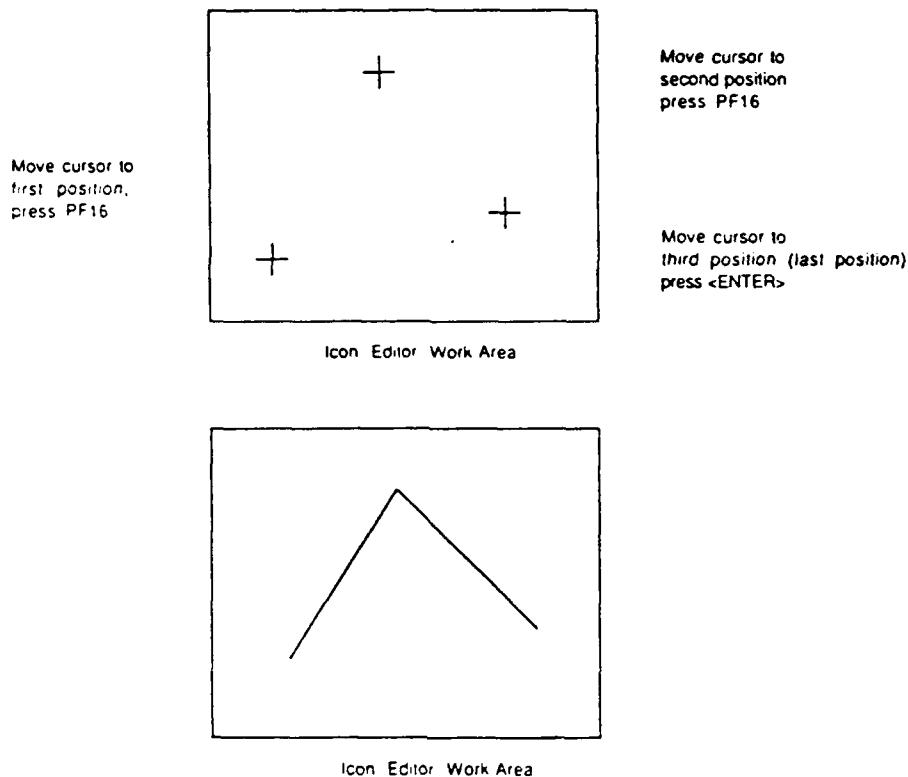


Figure 5-40 Polyline Primitive

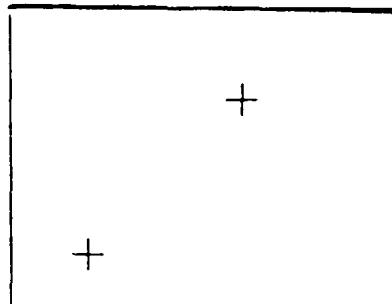
An icon using the circle, arc, or box primitives is created by placing the cursor on the minimum x, y position before pressing <PF16>; similarly, the cursor is placed on the maximum x, y position of the object before pressing <ENTER>. See Figure 5-41 for a representative example of the box formation.

5.5.6.4.2 Selection of an Object

An existing object is selected for modification or deletion by pressing <PF13> (for previous object) or <PF14> (for the next object) until the desired object is marked. A circle, arc, or box will have its width and depth marked with a dashed line. A polyline will be marked with a circle marker at each of its coordinates. The last coordinate of a polyline will be marked with a cross to show the default modification marker. Figures 5-42a thru 5-42c shows examples of the circle, box and polylines as they would appear when selected.

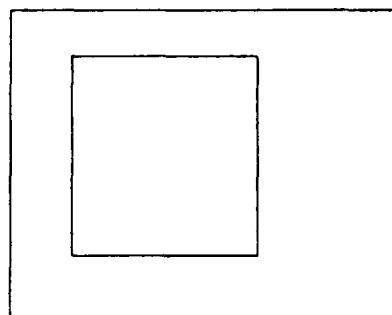
UM 620344400
30 September 1990

Place cursor at
minimum x,y
coordinate, press PF16



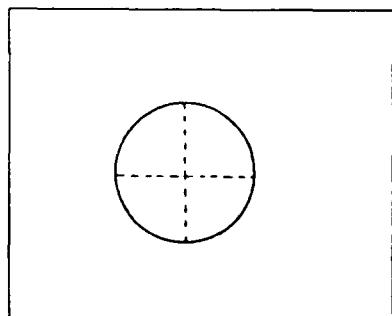
Place cursor at
maximum x,y
coordinate, press <ENTER>

Icon Editor Work Area

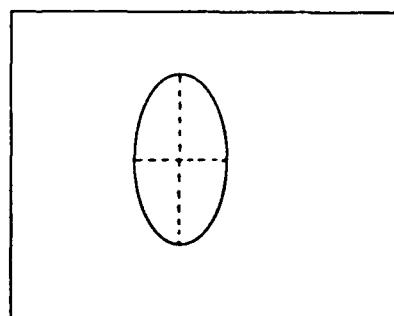


Icon Editor Work Area

Figure 5-41 Box Primitive



Icon Editor Work Area



Icon Editor Work Area

Figure 5-42 (a) Selected Circle

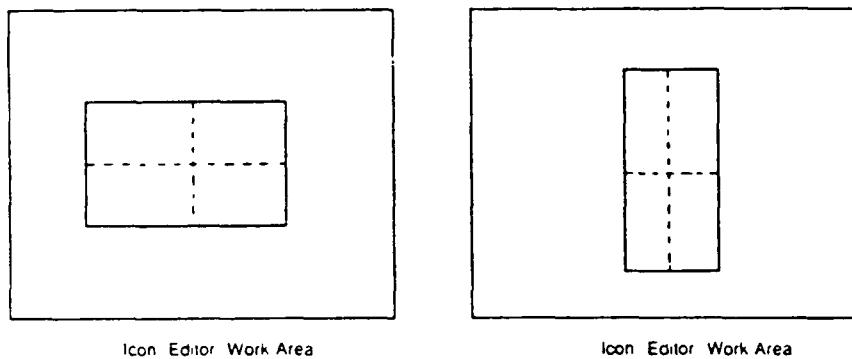


Figure 5-42 (b) Selected Box

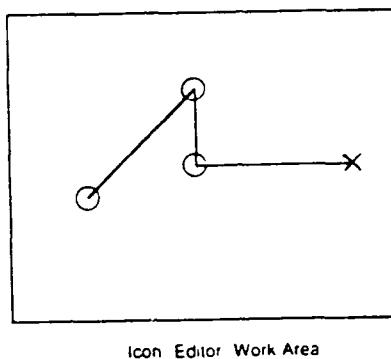


Figure 5-42 (c) Selected Polyline

5.5.6.4.3 Overlapping Objects

Objects in an icon can appear to overlap (Figure 5-43), or appear to be in front of, behind, or between other objects. Modifying such objects requires that the user imagine the objects are "STACKED" upon each other in some order. By selecting, and using the following commands, the user can place an object in the stack in a different position (behind, in front of, etc) other objects, as well as change its color, size, etc.

The user must first select an object, then the object can be put on the top of the stack, by pressing <PF15>. If the object is not explicitly put on the top of stack, it will be put back in its original position when another object is selected, or created, and when this editing mode is discontinued.

If an object is explicitly put on the top of the stack, it will not be returned to its original position, and it will remain in its current position on the stack.

If another object is selected after the currently selected object is explicitly placed on the top of stack, the currently selected object is unselected; it remains where it is on the stack and just below the object just selected.

If a new object is created after the currently selected object is explicitly put on the top of the stack,

- o the currently selected object is unselected
- o the old object remains where it is on the stack
- o and it is just below the object just created

For clarity, the user can blank-out all other objects in the stack, but the one selected. To blank-out the other objects, enter one of the modify icon modes and enter a 'Y' or 'y' to the prompt at the bottom of the change form (see Figure 5-45).

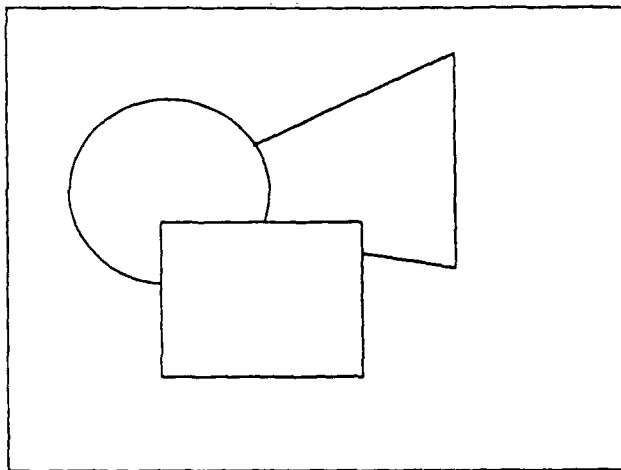


Figure 5-43 Overlapping Objects

5.5.6.4.4 Moving and Scaling of an Object

Two functions are provided to change the position and size of the four basic styles of objects. The change function command area (see also Figure 5-44) is located in the lower right corner of the screen.

To enter "move object's center function", the user tabs to move object and presses <ENTER>. When in this function, the screen resembles Figure 5-44-(A). Once in this function, to move the center of the selected object:

- o to move up press <PF5>
- o to move down press <PF6>
- o to move left press <PF7>
- o to move right press <PF8>

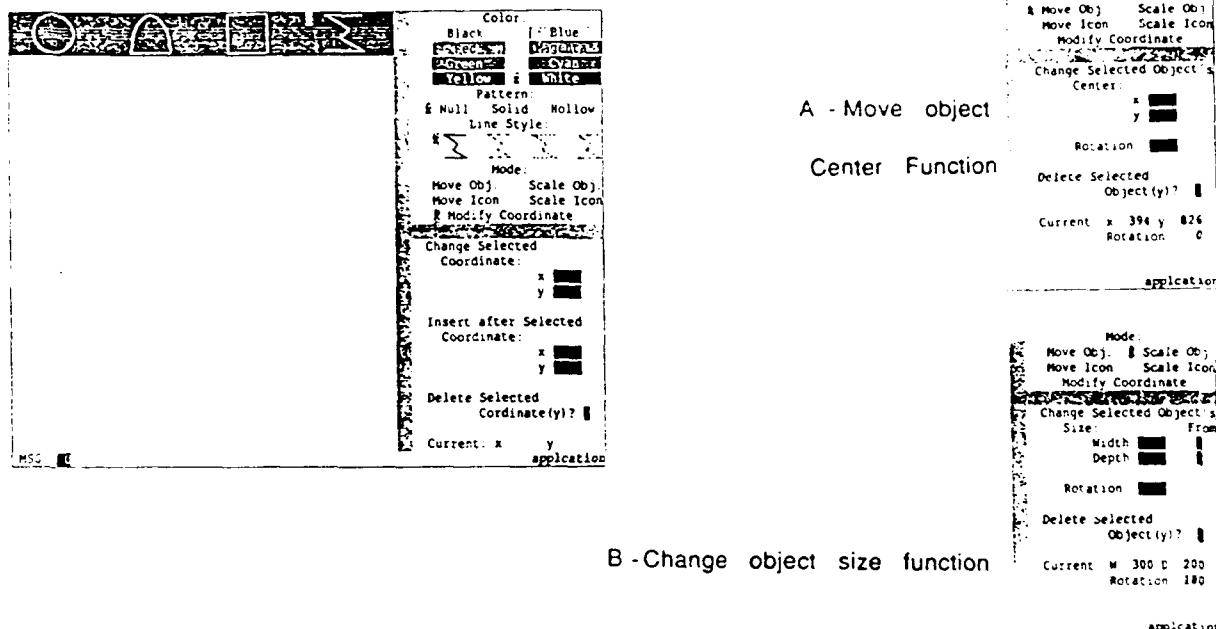


Figure 5-44 Moving or Scaling an Object
Showing Default Function (C)

Additionally to move the object, the new x,y of its center may be entered on the screen and the <ENTER> key pressed.

To enter "change object's dimensions function", the user tabs to scale object and presses <ENTER>. When in this function, the screen resembles Figure 5-44-(B). In this function, to change the dimensions of the selected object:

- o to increase the depth, press <PF5>
- o to decrease its depth, press <PF6>
- o to increase its width, press <PF7>
- o to decrease its width, press <PF8>

The new dimensions can also be entered on the screen and the <ENTER> key pressed.

To enter "move polyline's coordinates function", the user tabs to the move coordinate function and presses <ENTER>. When in this function, the screen looks like Figure 5-44-(C). In this function, to move the selected coordinate of a polyline:

- o to move it up, press <PF5>
- o to move it down, press <PF6>
- o to move it left, press <PF7>
- o to move it right, press <PF8>

The new x,y of coordinate of the polyline can also be entered on the screen and the <ENTER> key pressed.

To select a coordinate of a polyline selected for modification, the user repeatedly presses <PF9>, (for the previous coordinate) or, presses <PF10> (for the next coordinate), until the coordinate which is to be changed is marked as being selected.

5.5.6.4.5 Moving and Scaling an Icon

Two functions are provided to change the position and size of an entire icon. The change function command area (see also Figure 5-45) is located in the lower right corner of the screen.

To enter "move icon center function", the user tabs to move icon and presses <ENTER>. When in this function, the screen resembles Figure 5-45-(A). Once in this function, to move the center of the selected icon:

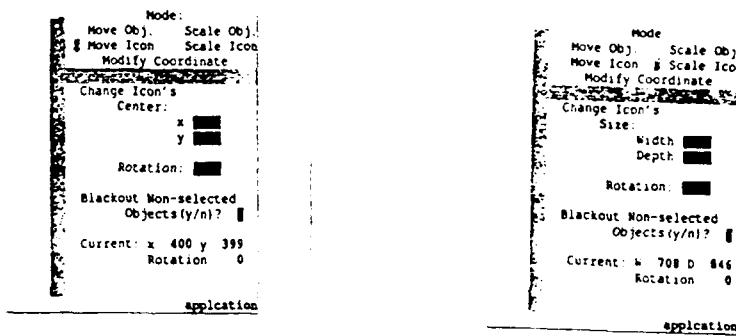
- o to move up press <PF5>
- o to move down press <PF6>
- o to move left press <PF7>
- o to move right press <PF8>

Additionally to move the icon, the new x,y of its center may be entered on the screen and the <ENTER> key pressed.

To enter "change icon dimensions function", the user tabs to scale icon and presses <ENTER>. When in this function, the screen resembles Figure 5-45-(B). In this function, to change the dimensions of the selected icon:

- o to increase the depth, press <PF5>
- o to decrease its depth, press <PF6>
- o to increase its width, press <PF7>
- o to decrease its width, press <PF8>

The new dimensions can also be entered on the screen and the <ENTER> key pressed.



A - Move icon center Function B - Change Icon size function

Figure 5-45 Moving and Scaling an Icon

Scale objects from point other than center:

Objects may be scaled from a point other than the center by specifying a point on the object to "hold stationary". The allowed points to hold are: top, bottom, left side, right side. Within the change object size function screen (see Figure 5-44-B) the user must specifying the hold-point after the width or depth (under From:). The valid entries are:

- o 'L' or 'l' after the new width if scaling from the left side
- o 'R' or 'r' after the new width if scaling from the right side
- o 'T' or 't' after the new depth if scaling from the top
- o 'B' or 'b' after the new depth if scaling from the bottom

5.5.6.4.6 Copy Selected Object

To copy the selected object, press <PF11>. The new object's center will be at the position which had been marked by the cursor when <PF11> was pressed.

5.5.6.4.7 Rotating

When in a modify object mode;

The selected object can be rotated about its center. To rotate a selected object, the user presses <PF12> which will rotate the object counter-clockwise 10 degrees. Also, an exact positive or negative integer value for the rotation can be entered on the screen and the <ENTER> key pressed.

When in a modify Icon mode;

The icon can be rotated about its center. To rotate an icon, the user presses <PF12> which will rotate the icon counter-clockwise 10 degrees. Also, an exact positive or negative integer value for the rotation can be entered on the screen and the <ENTER> key pressed.

5.5.6.4.8 Changing Color

To change the color or fill type used to draw objects the user tabs to the new color or fill pattern and presses <ENTER>. Changes must be made before object is drawn, to take effect. To change the color or fill pattern of an object, the user first selects the object. The selected object's color and fill become the current default. When these defaults are changed, the selected object's color and fill pattern is changed accordingly. The color white is the default.

5.5.6.4.9 Deleting an Object

To delete the currently selected object enter a 'y' into the field following 'Delete(y)?' and press <ENTER>.

5.5.6.4.10 Deleting a Polyline Coordinate

To delete a coordinate enter modify coordinate mode and enter a 'Y' or 'y' after delete coordinate prompt.

5.5.6.4.11 Inserting a Polyline Coordinate

To insert a coordinate enter modify coordinate mode and enter new coordinates into the modify coordinate form.

5.5.6.4.12 Changing Line Style

To change the line style, the user tabs to the line style as represented in the primitive area, and presses <ENTER>. The selected style will be marked with an X. The Solid style is the default style.

5.5.6.4.13 Keypad Help

Keypad Help is available by pressing <PF17>.

5.5.7 MF (Modify a Form)

Use this task to modify an existing form definition in the FDL source file shown in the "Form Source" field. This includes inserting new field definitions in the form and modifying any that the form already contains. You must enter the name of the form that you are going to modify and an edit mode to determine the editing style you want to use to modify the form.

The Command Entry format is: MF formname editmode<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Modify a Form, the name of the form as the "Form Name, S, F, or L as the "Edit Mode", and press the <ENTER> key. For example:

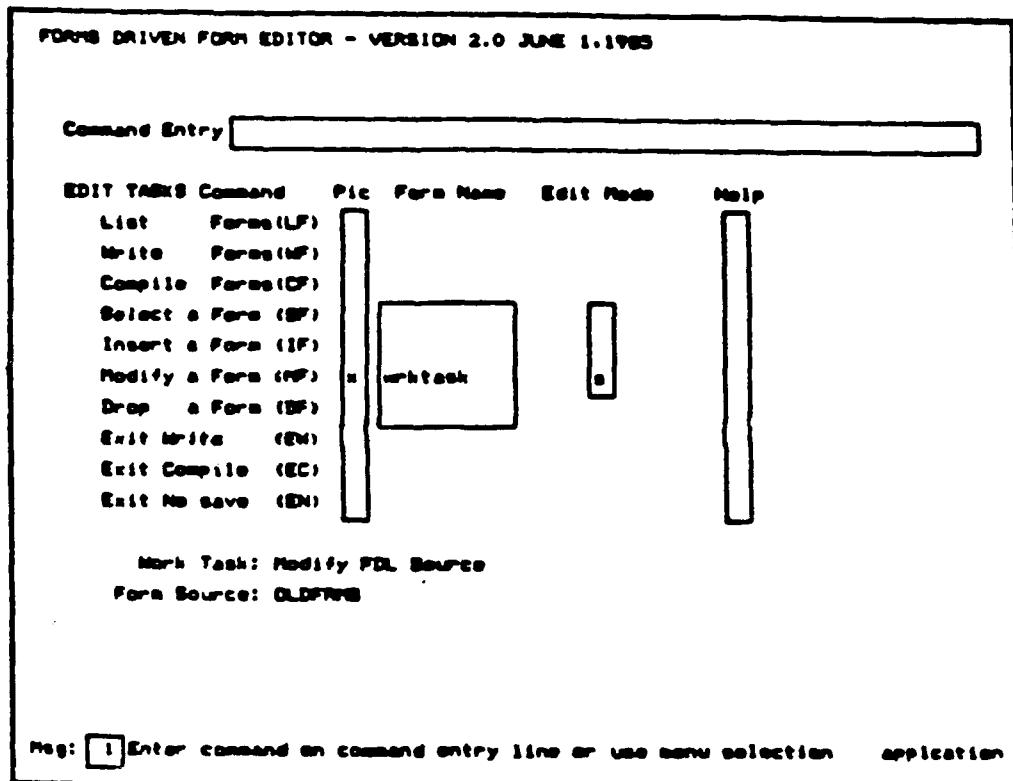


Figure 5-46 Choosing MF Using Menu Selection

The next screen displayed depends on the Edit Mode you choose. The next three sections explain how to modify a form using the Single Field, Form, and Layout edit modes.

5.5.7.1 Using Single Field Mode to Modify a Form

When you execute the MF edit task with the "S" edit mode, the next screen displayed is:

FIELD EDIT MODE		For FDL File CLOTHES	
Task	<input type="text" value="C"/>	Form	<input type="text"/>
Field	<input type="text"/>	Form Newform	<input type="text"/>
Type	<input type="text"/>	Size	<input type="text"/> 0 by 0
Direct	<input type="text"/>	Background	<input type="text"/> BLACK
Get FDL	<input type="text"/>	Prompt	<input type="text"/> EXAMPLE FORM
		Row	<input type="text"/> 1 Column 10
-- Field Information --			
REQUIRED:		OPTIONAL:	
FIELD	<input type="text"/>	Display	<input type="text"/>
Row	<input type="text"/>	Actual	<input type="text"/>
Size	<input type="text"/>	Direction	<input type="text"/>
Display/	<input type="text"/>	Spacing	<input type="text"/>
Background	<input type="text"/>	Prompt	<input type="text"/>
		Pos	<input type="text"/> Row Col
ITEM ONLY:			
Help	<input type="text"/>	MIN Value	<input type="text"/>
Value	<input type="text"/>	MAX Value	<input type="text"/>
Justify	<input type="text"/>		<input type="text"/>
Case	<input type="text"/>	Data Type	<input type="text"/>
		Enter/Fill	<input type="text"/>
Msg: <input type="text"/> Enter task and field to be acted on. application			

Figure 5-47 Single Field Mode Screen

If you want to modify information in the FORM area of the screen, make the desired changes, enter "F" in the "TASK" field, and press the <ENTER> key.

You can insert new fields in the form by executing the INSERT or COPY field task as described in section 5.5.6.1.

You can delete fields from the form using the DROP field task. To execute the DROP task, enter "D" in the "TASK" field, use the "Field" or "Type" and "Direct" fields to identify the field you want to delete, and press the <ENTER> key or press the <NEXT> or <PREV> key after entering "D" in the "TASK" field to identify the field to be dropped. The field is displayed in the field information template and the message "Field on screen dropped!" is displayed in the message line. For example:

FIELD EDIT MODE **For FDL File OLDPRPS**

Task	FIELD	Type	FORM REPORT
Field		Size	0 by 0
Type		Background	BLACK
Direct		Prompt	EXAMPLE FORM
Get FDL	Form	Row	1 Column 10

— Field Information —

REQUIRED:		OPTIONAL:	
FIELD	FIELD	Type	Field
Row	3	Column	15
Size	12	By	1
Display/	INPUT	Actual	Repetition
Background		Direction	(-)
		Prompt	Item 1:
		Pos	LT Row 3 Col 7

ITEM ONLY:

Help		
Value		
Justify	Data Type	MIN Value
Case	Enter/Fill	MAX Value

Reg: Field on screen dropped! application

Figure 5-48 Using Single Field Mode to Delete a Field

If you decide that you do not want this field deleted, execute the INSERT task to cancel the DROP task by immediately entering "I" in the task field and pressing the <ENTER> key.

You can modify any of the fields that the form currently contains using the MODIFY field task. To execute the MODIFY task, enter "M" in the "TASK" field, use the "Field" or "Type" and "Direct" fields to identify the field to be modified and press the <ENTER> key. You can also press the <NEXT> or <PREV> key after entering "M" in the "TASK" field to identify the field to be modified. The field will be displayed in the field information template and the message "Make your modifications now" is displayed in the message line. At this time you make any changes to the field information and press the <ENTER> key when you have the field modified as you want. The message "Field successfully modified" is displayed in the message line. If you decide that you do not want to modify this field after all, press the <QUIT> key to cancel the task and restore values if you have already made changes. If you have not made any changes in the field information, you can press the <ENTER> or the <QUIT> key to cancel the task.

You can review any of the fields that the form currently contains using the SELECT field task. To execute the SELECT field task, enter "S" in the "TASK" field, use the "Field" or "Type" and "Direct" fields to identify the field to be reviewed

and press the <ENTER> key. You can also press the <NEXT> or <PREV> key after entering "S" in the "TASK" field to identify the field to be reviewed. The field will be displayed in the field information template but you cannot change any of the information. You can continue reviewing fields on the form as described in section 5.5.5.1. If you find that a field needs modifying or that you want to use a field to help you define a new one, you can then execute the MODIFY or INSERT field task.

5.5.7.2 Using Form Mode to Modify a Form

When you execute the MF edit task with the "F" edit mode, the next screen displayed is:

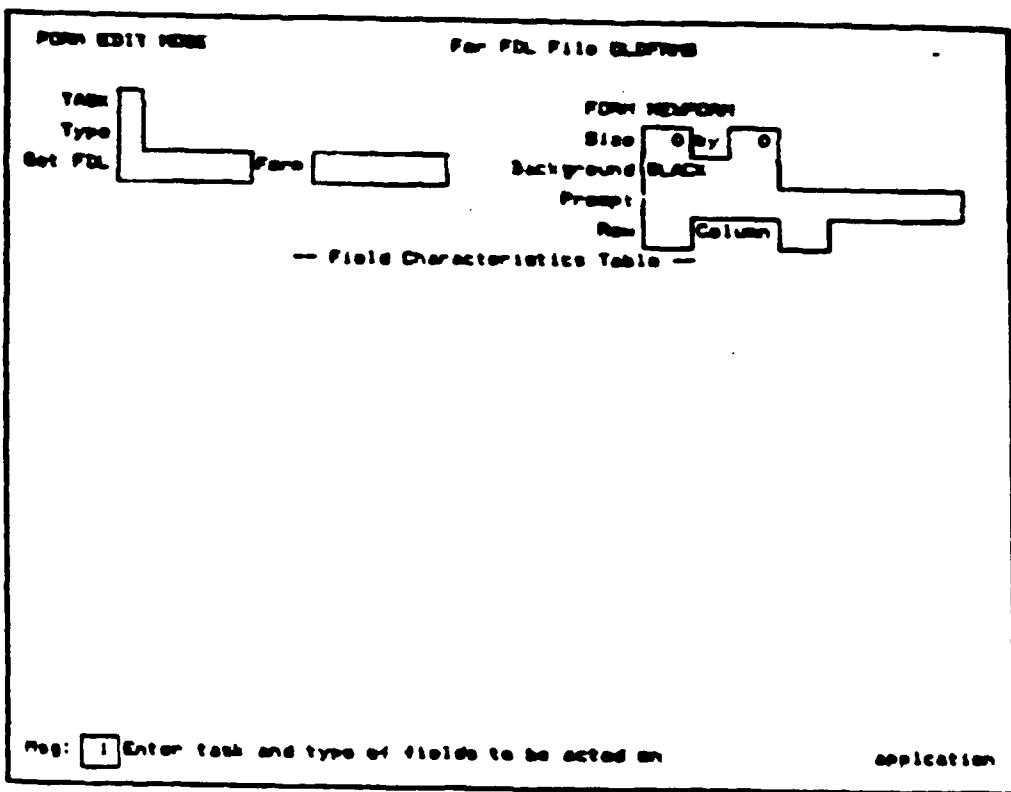


Figure 5-49 Form Mode Screen

If you want to modify information in the FORM area of the screen, make the desired changes in these fields, enter "F" in the "TASK" field, and press the <ENTER> key.

You can define new fields for the current form using the INSERT or COPY field task as explained in section 5.5.6.2.

You can modify existing fields on the form using the MODIFY field task. To execute the MODIFY task, enter "M" in the "TASK" field, "I", "F", "W", or "A" in the "Type" field to identify the type of fields you want to modify, and press the <ENTER> key. The FIELD CHARACTERISTICS TABLE will be displayed with the identified fields and the message "Make your modifications" now is displayed in the message line. At this time you can modify any of the entries in the table and press the <ENTER> key to enter the changes. If you decide not to make any modifications or you want to restore the values you changed, press the <QUIT> key instead of the <ENTER> key to cancel the MODIFY task.

You can review any of the fields that the form currently contains using the SELECT field task. To execute the SELECT task, enter "S" in the "TASK" field, "I", "F", "W", or "A" in the "Type" field to identify the type of fields you want to review, and press the <ENTER> key. The fields will be displayed in the FIELD CHARACTERISTICS TABLE but you cannot change any of

the information. If you decide that the field information needs modifying or that you want to use a field to help you define a new one, you can then execute the MODIFY or INSERT field task.

You can delete fields from the form using the DROP field task. To execute the DROP task, enter "D" in the "TASK" field, "I", "F", "W", or "A" in the "Type" field to identify the type of fields you want to delete, and press the <ENTER> key. The message "Mark fields to be dropped with an '*'!" is displayed in the message line and the FIELD CHARACTERISTICS TABLE is displayed with a "DELETE" column. For example:

FORM EDIT MODE

For FDL File Options

Task	C
Type	I
Set FDL	Form
Form	
Background	Black
Size	0 by 0
Prompt	EXAMPLE FORM
Row	1 Column 10

— Field Characteristics Table —

Field Name	T	Row	Col	Size	Display	Def Act	D Sp	Prompt	Pos
FIELD1	I	3	15	15	1 INPUT			Item 1:	LT
FIELD2	I	4	15	15	1 INPUT			Item 2:	LT
FIELD3	W	7	15	15	8 BLACK			Window 1	TL

Msg: Mark fields to be dropped with an '*'!

application

Figure 5-50 Using Form Mode to Delete a Field

Mark the field(s) to be dropped by entering an "*" in the appropriate delete column position(s) and press the <ENTER> key. The number preceding the message line tells you how many fields were dropped and a message verifying the last field deleted is displayed in the message line. You can use the <MESSAGE QUEUE> key to verify that any other marked fields were successfully deleted.

5.5.7.3 Using Layout Mode to Modify a Form

When you execute the MF edit task with the "L" edit mode, the form is displayed using the special layout symbols described in section 5.5.6.3. For example:

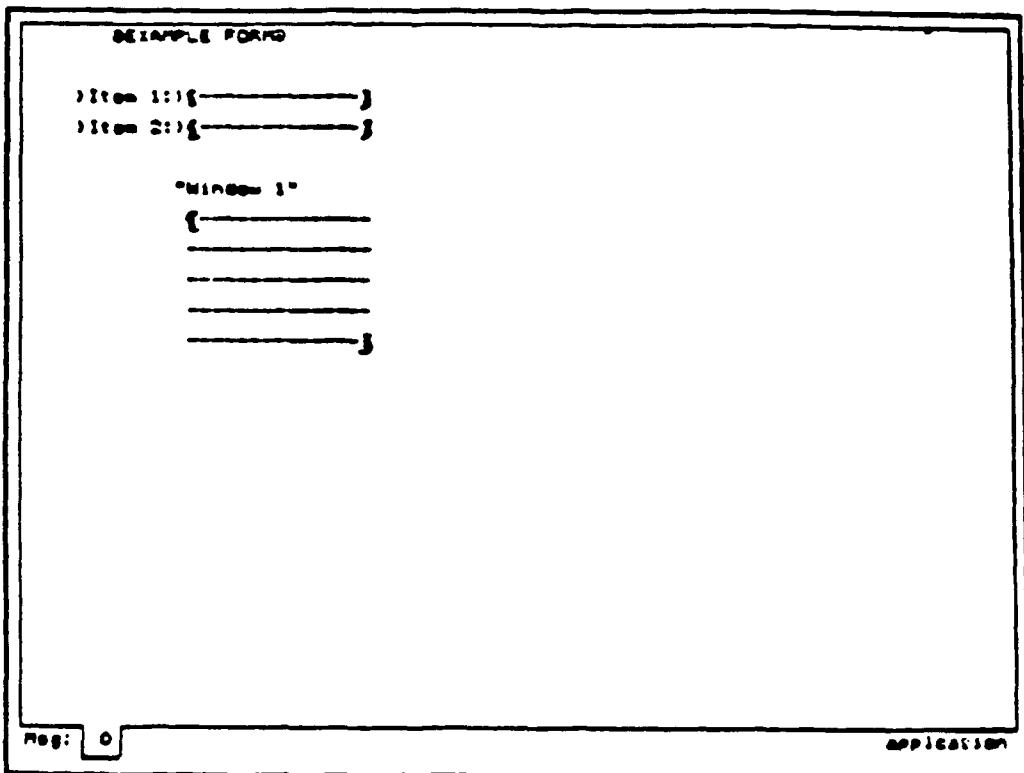


Figure 5-51 Modifying a Form in Layout Mode

This screen allows you to modify the form by graphically positioning fields and prompts as described in section 5.5.6.3.

5.5.7.4 Using Icon Mode to Modify a Form

When you execute the MF edit task with the "I" edit mode, the icon form is displayed as described in section 5.5.6.4, with the exception of the defaults which when modifying an icon are obtained from the initially selected object - the object on top of the stack. For example:

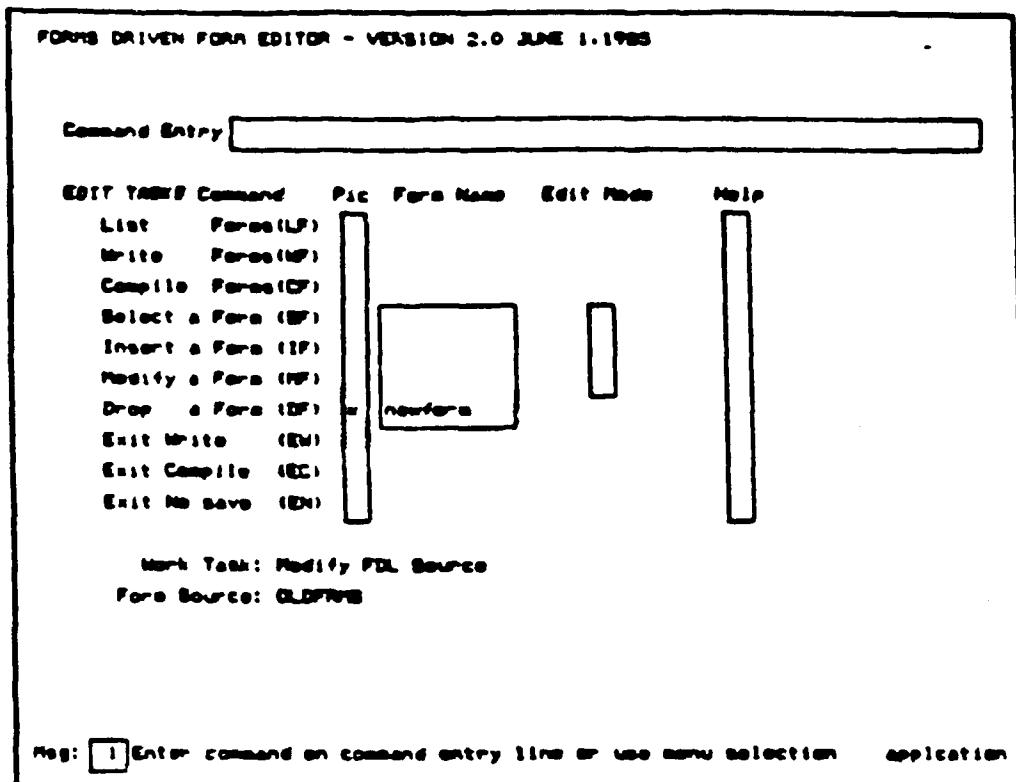


Figure 5-52 Modifying an Icon in Icon Mode

This screen allows you to modify the icon as described in section 5.5.6.4

5.5.8 DF (Drop a Form)

Use this task to drop (delete) a form definition from the FDL source file displayed in the "Form Source" field. You must enter the name of the form that you want to delete.

The command entry format is: DF formname<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Drop a Form, the name of the form as the "Form Name", and press the <ENTER> key. For example:

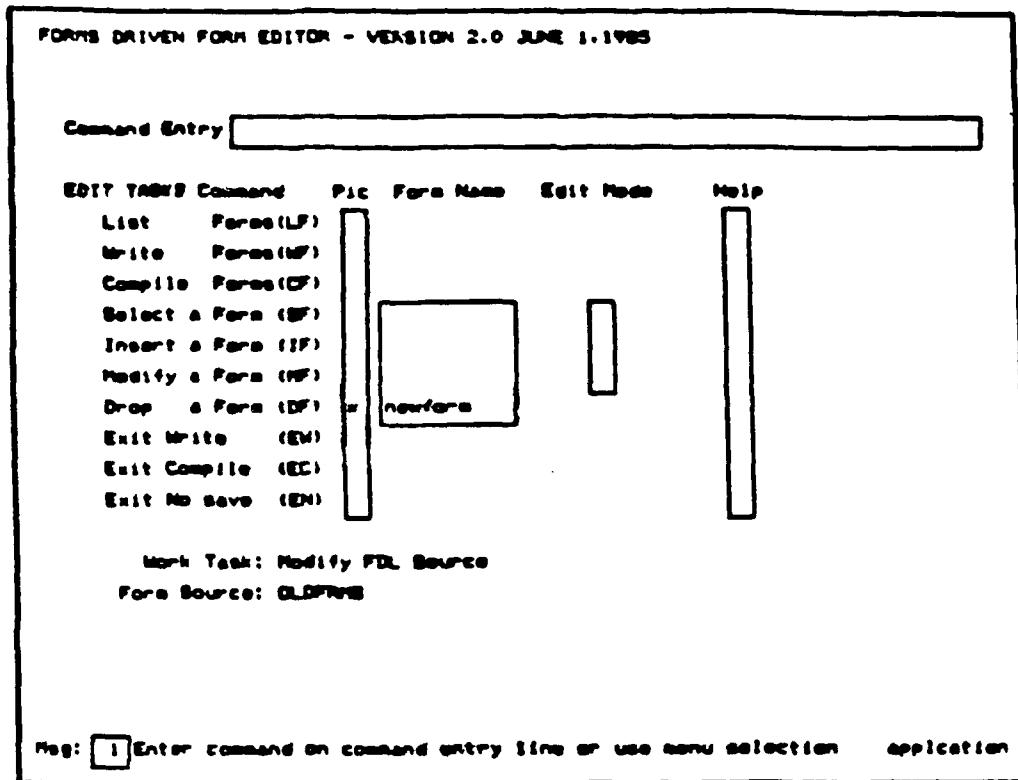


Figure 5-53 Choosing DF Using Menu Selection

When this task is executed, the message "Drop was successful" is displayed in the message line and you remain at the Edit Task level. Note that if the FDL source file has been compiled, the FD file for the form is not deleted. You delete this file using the DC Work Task.

5.5.9 EW (Exit Write)

Use this task to execute the WF (Write Form) Edit Task and exit the FDFE. This task saves the current FDL source file without compiling it and generating the FD files.

The Command Entry format is: EW<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Exit Write and press the <ENTER> key.

5.5.10 EC (Exit Compile)

Use this task to compile the form definition(s) you are currently working on and then exit the FDFE.

The Command Entry format is: EC<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Exit Compile and press the <ENTER> key.

If there are syntax errors, the message "Form compile errors, review errors and correct" is displayed in the message line. The errors are stored in the message queue which can be displayed using the <MESSAGE QUEUE> key. When there are no syntax errors, the form definition(s) will be saved in the FDL source file shown in the "Form Source" field, the compiled form(s) will be saved as FD file(s), and you will exit the FDFE.

5.5.11 EN (Exit No save)

Use this task to terminate your current FDFE edit session without saving or compiling anything.

The Command Entry format is: EN<ENTER>

To execute this task using menu selection, enter any nonblank character in the "Pic" field for Exit No save and press the <ENTER> key.

5.6 Limited Edit Task Screen

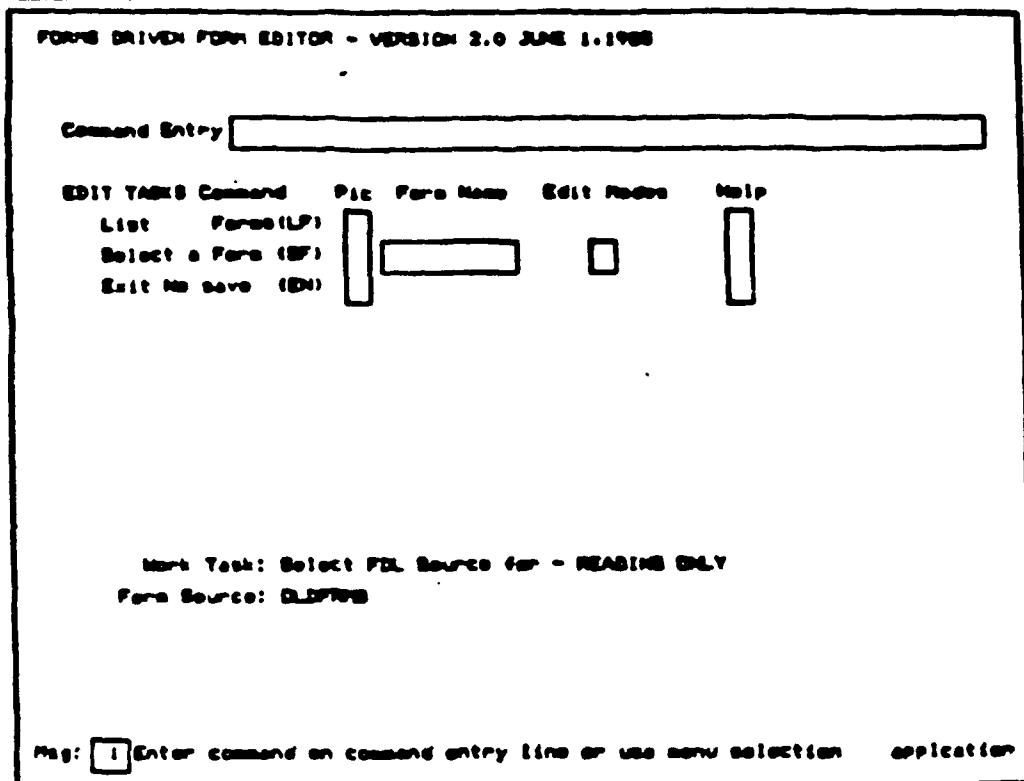


Figure 5-54 Limited Edit Task Screen

This screen is the next screen displayed when you execute the SS work task. It is a list of the READ ONLY FDFE functions available for form definitions contained in the specified FDL source file. How to execute these tasks is explained in section 5.5.

SECTION 6

FDL COMPILER - FLAN

The compiler for the Form Definition Language is FLAN. FLAN must be used to convert form definitions into a format understood by the Form Processor. FLAN reads an FDL source file and creates a separate compiled form for each form definition contained in the source file. A compiled form is referred to as an FD file.

6.1 Executing FLAN

FLAN is available as an application in the IISS environment or as a batch program on the host system.

6.1.1 FLAN in the IISS Environment

To execute FLAN in the IISS environment, enter FLAN as the FUNCTION on the IISS Function Screen. When you have successfully accessed FLAN, the following display appears on your terminal screen:

```
+-----+
| IISS Form Definition Language Compiler Release 2.0
+-----+
| Form Definition Language File Name: _____
|
|
|
|
+-----+
MSG: 0                                     application
```

Enter the name of the FDL source file you want to compile. ".FDL" is the default extension. The compiled forms (FD files) are written to the location defined by the UI symbolic name IISSULIB.

6.1.2 FLAN as a Batch Program

FLAN is also available as a batch program outside the IISS and UIS environment. When you use this version of FLAN, you are prompted for the FDL source file name. To execute this version of FLAN, enter the appropriate executable name at the system prompt. This name should be defined as FLAN also.

6.2 FLAN Error Messages

Error messages in FLAN consist of the line number of the error, the level of the error, and an error message which briefly describes the error. There are three levels of errors:

- o **Warnings:** The error does not prevent the creation of a .FD file although the form may not display as expected.
- o **Errors:** The error is sufficiently serious to prevent the creation of a .FD file but syntax checking will continue from this point.
- o **Fatals:** The error is very serious and prevents the creation of a .FD file and further compilation.

6.2.1 Warning Messages

1. form <form name> too wide for standard screen.
2. form <form name> too long for standard screen.
3. help message too long, truncated.
4. string too long.

6.2.2 Error Messages

1. <number of> errors detected.
2. unable to open file <filename> for form <form name>.
3. value too big for field.
4. size not specified or invalid.
5. no display attribute specified.

6. field <field name> referenced in <field type> <field name> [prompt] not defined.
7. circular reference in location of <field type> <field name> [prompt].
8. overlap between <field type> <field name> [prompt] and <field type> <field name> [prompt].
9. <field type> <field name> [prompt] off top of screen.
10. <field type> <field name> [prompt] off left of screen.
11. unknown background attribute: <attribute name>.
12. must specify relative field name.
13. duplicate field name: <field name>.
14. duplicate display attribute specified.
15. domain only legal for items.
16. duplicate justification specified.
17. duplicate case specified.
18. duplicate minimum specified.
19. duplicate maximum specified.
20. help only legal for items.
21. duplicate help specified.
22. duplicate value specified.
23. duplicate size specified.
24. unterminated string.
25. value only legal for items.
26. unknown function <function name>.
27. invalid argument for INDEX.
28. form <form name> too narrow: fields extend to column <integer>.
29. form <form name> too short: fields extend to row <integer>.
30. yacc stack overflow.
31. syntax error.

6.2.3 Fatal Messages

1. out of memory.
2. unable to open input file <file name>.
3. unterminated comment.
4. internal error: corrupt poslst.

SECTION 7

FORM TOOLS

7.1 Generating Include Members

After compiling a form definition, you can use MAKINC to generate an include file containing the data structure which corresponds to the form. This data structure can then be used in application programs which get data from or put data to the form using the FP routines GDATA and PDATA. The following table describes the series of prompts and your responses.

System Prompt	Your Response
Your system prompt	Log on to the system
Your system prompt	Invoke MAKINC
AVAILABLE LANGUAGES:	
0 - C	
1 - COBOL	
2 - PL/I	
ENTER LANGUAGE:	1
	For this example we will use COBOL
FORM NAME:	MM
	You can enter the name of any form which has been compiled. For this example we will use the Message Management form whose definition is shown in Section 8.
FORM NAME:	<END OF FILE>
	This sequence is usually CONTROL/Z. You can make include files for more than one form, so the prompt for form name is repeated. When you are finished making include files, enter the <END OF FILE> sequence for your system.

Your include files are created on your current directory with the same name as the form and a type (or extension) of INC on a VAX host. For C programs it is recommended that the file type be changed to H. If you want to view the files, you can use any available editor or display command. For this example, we use the type command as follows:

```
$ TYPE MM.INC
01  MM-FORM-DATA.
02  MBASE PIC X(5).
02  MSGLIN-FORM-DATA OCCURS 10 TIMES.
03  NUMBER PIC X(5).
03  MSGNAM PIC X(8).
03  MSGDES PIC X(60).
```

This is your COBOL include file.

NOTE that the UI symbolic name IISSILIB points to the location of the source for COBOL include files and you may want to move any COBOL include files generated by MAKINC to this directory.

SECTION 8

SAMPLE FORM DEFINITION

This is the FDL source for the form used in the Message Management program. Figure 8-1 shows the form that is defined.

Figure 8-1 Sample Form

```
CREATE FORM mm
    PROMPT CENTER AT 1 20
        "ERROR MESSAGE DEFINITION SCREEN"
    PROMPT CENTER AT 2 20
        "-----"
    PROMPT AT 6 2 "NUMBER"
    PROMPT AT 6 10 "NAME"
    PROMPT AT 6 20 "DESCRIPTION"
```

```
ITEM mbase
DISPLAY AS input
HELP "Enter the error message's base number"
PROMPT AT 4 2 "Message Base Number:"
AT 4 25
SIZE 5

FORM msglin (10 V WITH 0 SPACES)
DISPLAY AS output
AT 7 2
SIZE 78

CREATE FORM msglin

ITEM number
DISPLAY AS output
AT 1 2
SIZE 5

ITEM msgnam
DISPLAY AS input
HELP "This value is the name of the error"
AT 1 9
SIZE 8

ITEM msgdes
DISPLAY AS input
HELP "Description of the error"
AT 1 19
SIZE 60
```